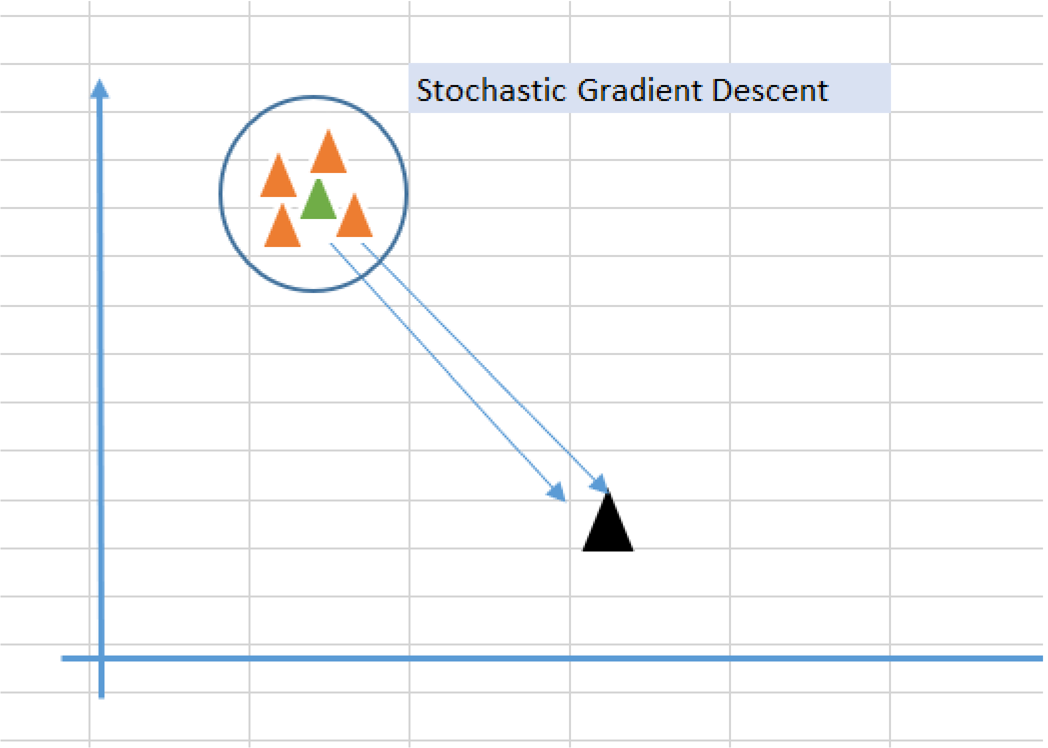
Stochastic Gradient Descent

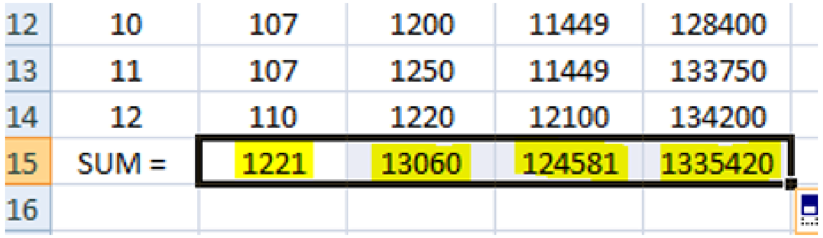


Data Sets:

gd_lr.csv

cs2m_scaled.csv

Old is Gold!



12	10	107	1200	11449	128400	
13	11	107	1250	11449	133750	
14	12	110	1220	12100	134200	
15	SUM =	1221	13060	124581	1335420	
16						

$$SS_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$$

$$SS_{xy} = 1335420 - \frac{1221 * 13060}{12} = 6565$$

$$SS_{xx} = \sum x^2 - \frac{(\sum x)^2}{n} = 124581 - \frac{(1221)^2}{12} = 344.25$$

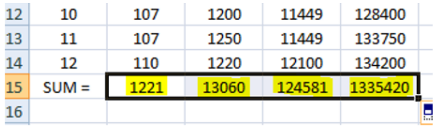
This is called Regression Coefficient

$$\boldsymbol{b_1} = \frac{SS_{xy}}{SS_{xx}} = \frac{6565}{344.25} = 19.07$$

This is called Intercept

$$\boldsymbol{b_0} = \frac{\sum y}{n} - b_1 \frac{\sum x}{n} = \frac{13060}{12} - 19.07 \frac{1221}{12} = -852.08$$

Estimate of Constant



12	10	107	1200	11449	128400
13	11	107	1250	11449	133750
14	12	110	1220	12100	134200
15	SUM =	1221	13060	124581	1335420
16					

$$SS_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$$

$$SS_{xy} = 1335420 - \frac{1221 * 13060}{12} = 6565$$

$$SS_{xx} = \sum x^2 - \frac{(\sum x)^2}{n} = 124581 - \frac{(1221)^2}{12} = 344.25$$

This is called Regression Coefficient

This is called Intercept

$$\boldsymbol{b_1} = \frac{SS_{xy}}{SS_{xx}} = \frac{6565}{344.25} = 19.07$$

$$\boldsymbol{b_0} = \frac{\sum y}{n} - b_1 \frac{\sum x}{n} = \frac{13060}{12} - 19.07 \frac{1221}{12} = -852.08$$

$$y_{predicted} = a + bx_i$$

$$S = \sum_{i=1}^{n} \left(y_i - (a + bx_i)\right)^2$$

By opening the bracket, signs will change

$$S = \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

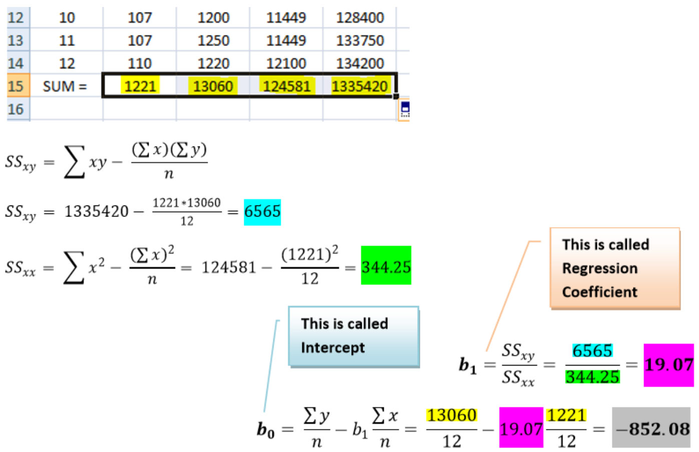
Find 1st partial derivative of S with respect to a

$$\frac{\partial}{\partial a} = 0 = \frac{\partial}{\partial a}\left[\sum_{i=1}^{n} (y_i - a - bx_i)^2\right]$$

Applying chain rule, derivative of x^n with respect to x is $= nx^{n-1}$

$$0 = \sum_{i=1}^{n}\left[2(y_i - a - bx_i) \times \frac{\partial}{\partial a}(y_i - a - bx_i)\right]$$

Estimate of Constant



12	10	107	1200	11449	128400
13	11	107	1250	11449	133750
14	12	110	1220	12100	134200
15	SUM =	1221	13060	124581	1335420
16					

$$SS_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$$

$$SS_{xy} = 1335420 - \frac{1221 * 13060}{12} = 6565$$

$$SS_{xx} = \sum x^2 - \frac{(\sum x)^2}{n} = 124581 - \frac{(1221)^2}{12} = 344.25$$

This is called Regression Coefficient

$$\boldsymbol{b_1} = \frac{SS_{xy}}{SS_{xx}} = \frac{6565}{344.25} = \mathbf{19.07}$$

This is called Intercept

$$\boldsymbol{b_0} = \frac{\sum y}{n} - b_1 \frac{\sum x}{n} = \frac{13060}{12} - 19.07 \frac{1221}{12} = \mathbf{-852.08}$$

$$We\ bring\ 2\ left\ side, \frac{0}{2}\ is\ again\ 0$$

$$0 = \sum_{i=1}^{n} \left[(y_i - a - bx_i) \times \frac{\partial}{\partial a} (y_i - a - bx_i) \right]$$

$$0 = \sum_{i=1}^{n} \left[(y_i - a - bx_i) \times \frac{\partial}{\partial a} y_i - \frac{\boldsymbol{\partial}}{\boldsymbol{\partial a}} \boldsymbol{a} - \frac{\partial}{\partial a} bx_i \right]$$

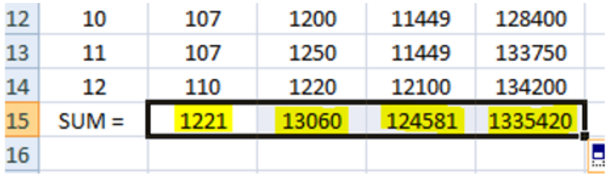
$$0 = \sum_{i=1}^{n} [(y_i - a - bx_i) \times \ 0 - \mathbf{1} - 0]$$

$$0 = \sum_{i=1}^{n} [(y_i - a - bx_i) \times \ -\mathbf{1}]$$

$$We\ bring - 1\ left\ side, \frac{0}{-1}\ is\ again\ 0$$

$$0 = \sum_{i=1}^{n} [(y_i - a - bx_i)]$$

Estimate of Constant



12	10	107	1200	11449	128400
13	11	107	1250	11449	133750
14	12	110	1220	12100	134200
15	SUM =	1221	13060	124581	1335420
16					

$$SS_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$$

$$SS_{xy} = 1335420 - \frac{1221 * 13060}{12} = 6565$$

$$SS_{xx} = \sum x^2 - \frac{(\sum x)^2}{n} = 124581 - \frac{(1221)^2}{12} = 344.25$$

This is called Regression Coefficient

$$\boldsymbol{b_1} = \frac{SS_{xy}}{SS_{xx}} = \frac{6565}{344.25} = 19.07$$

This is called Intercept

$$\boldsymbol{b_0} = \frac{\sum y}{n} - b_1 \frac{\sum x}{n} = \frac{13060}{12} - 19.07 \frac{1221}{12} = -852.08$$

Taking summation inside the bracket

$$0 = \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a - b \sum_{i=1}^{n} x_i$$

Summation of a , n times is na

$$0 = \sum_{i=1}^{n} y_i - na - b \sum_{i=1}^{n} x_i$$

Bringing $-na$ left side

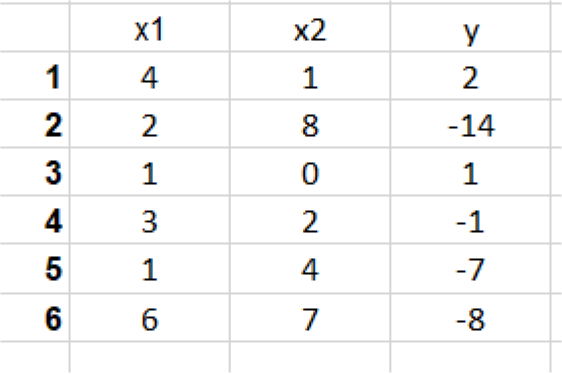
$$na = \sum_{i=1}^{n} y_i - b \sum_{i=1}^{n} x_i$$

$$a = \frac{\sum_{i=1}^{n} y_i - b \sum_{i=1}^{n} x_i}{n}$$

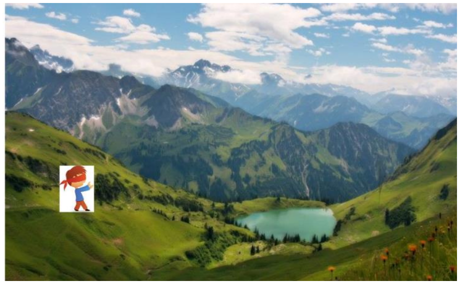
$$a = \frac{\sum_{i=1}^{n} y_i}{n} - b \frac{\sum_{i=1}^{n} x_i}{n}$$

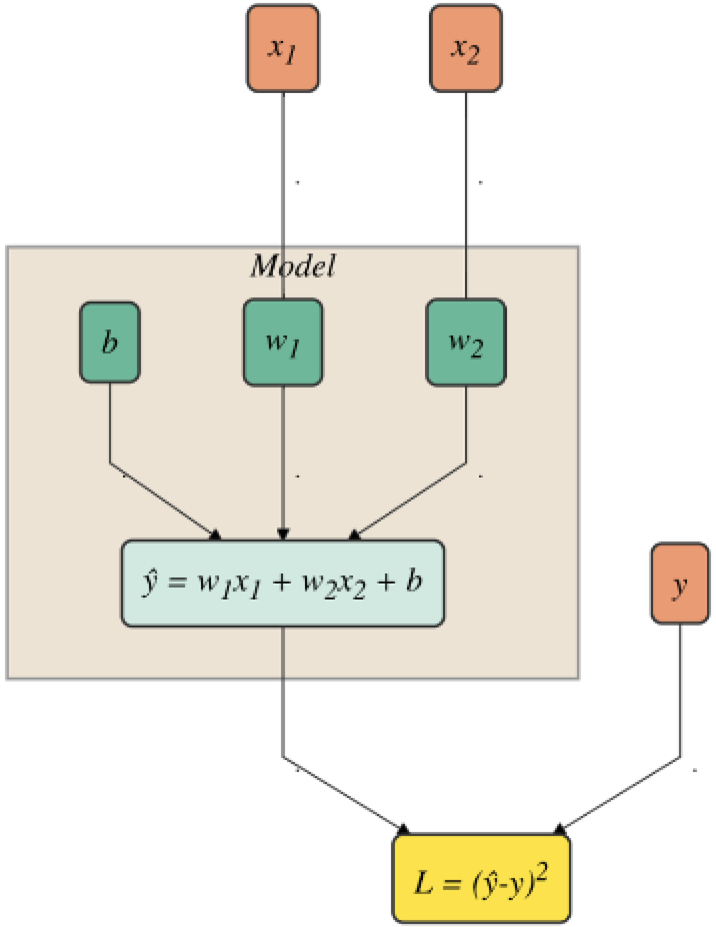
Gradient Descent

Gradient Descent: Data and Model

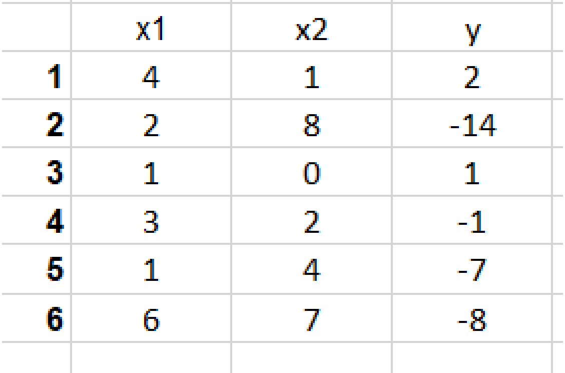


	x1	x2	y
1	4	1	2
2	2	8	-14
3	1	0	1
4	3	2	-1
5	1	4	-7
6	6	7	-8

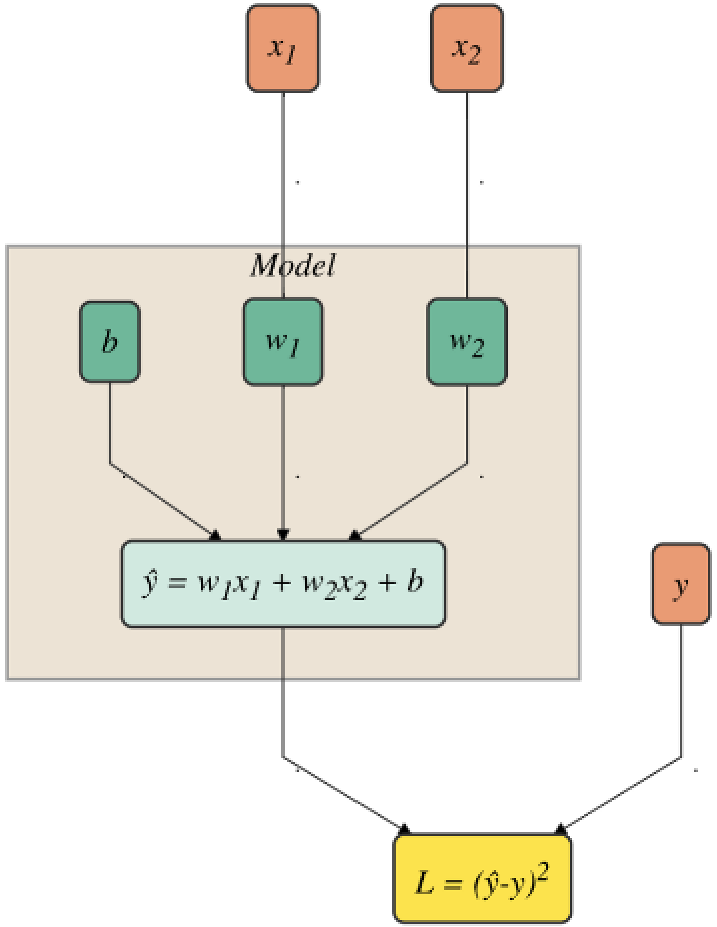




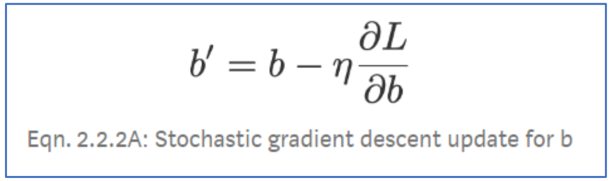
Data and Model



	x1	x2	y
1	4	1	2
2	2	8	-14
3	1	0	1
4	3	2	-1
5	1	4	-7
6	6	7	-8



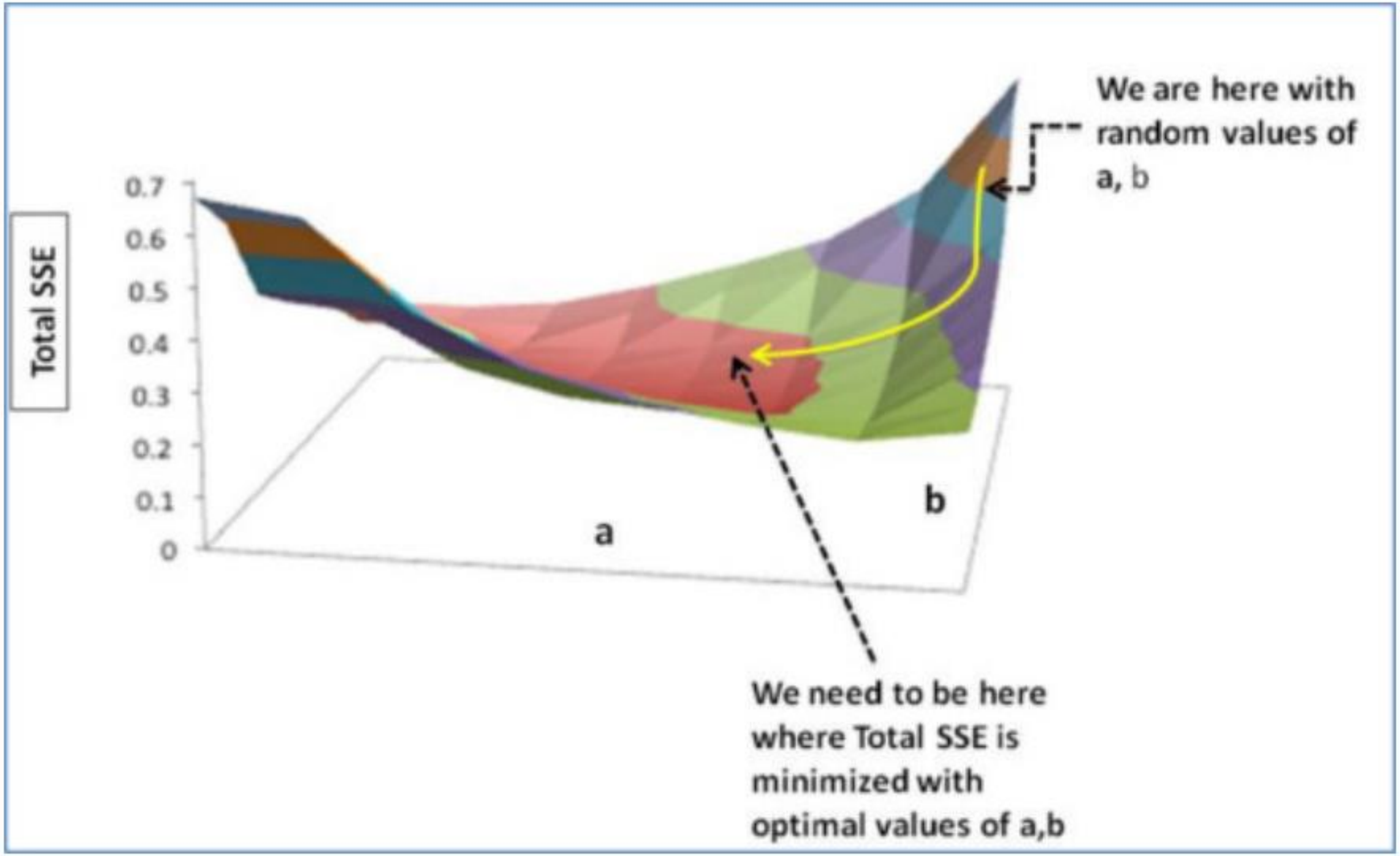
Batch, initial weights, Loss Function, Learning Rate, Partial Differentiation



$$b' = b - \eta \frac{\partial L}{\partial b}$$

Eqn. 2.2.2A: Stochastic gradient descent update for b

Step 3: Adjust the weights with the gradients to reach the optimal values where SSE is minimized



We are here with random values of a, b

We need to be here where Total SSE is minimized with optimal values of a,b

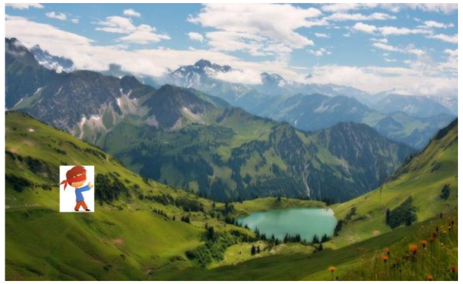
Weight Updation

$$Updated\ weight\ a_{n+1} = old\ weight\ (a_n) - learning\ rate \times \frac{\partial}{\partial \boldsymbol{a}} SSE$$

First update in **a**

$$Updated\ weight\ a_{n+1} = 0.45 - 0.01\ \times\ 3.30 = \mathbf{0.417 = 0.42}$$

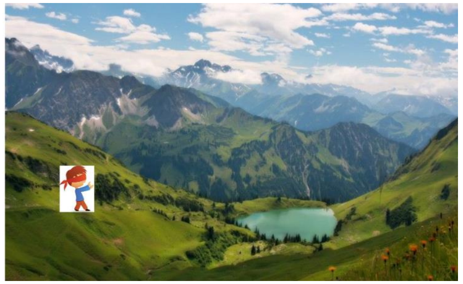
$$Updated\ weight\ b_{n+1} = old\ weight\ (b_n) - learning\ rate \times \frac{\partial}{\partial \boldsymbol{b}} SSE$$

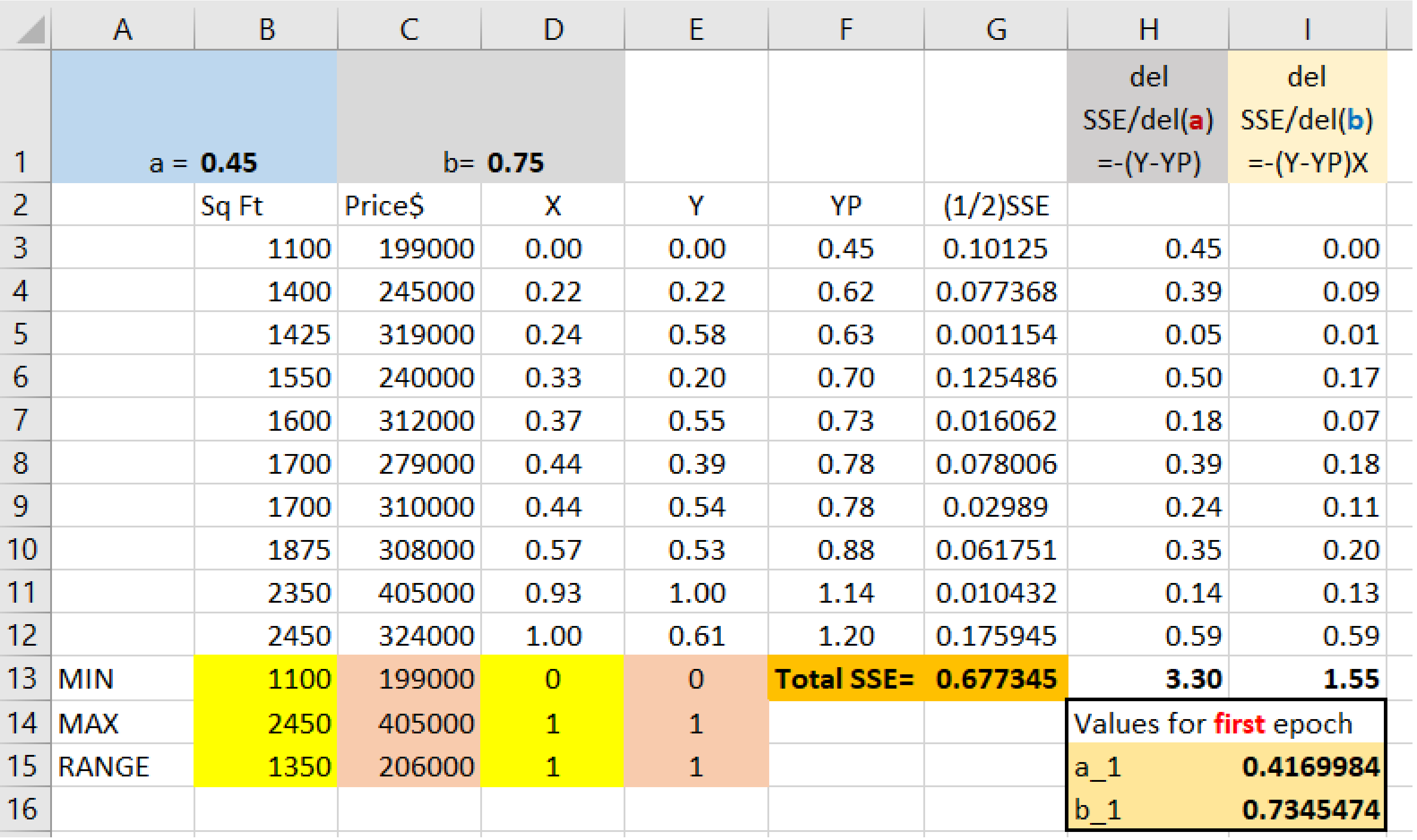


First update in **b**

$$Updated\ weight\ b_1 = 0.75 - 0.01\ \times\ 1.55 = \mathbf{0.7345 = 0.73}$$

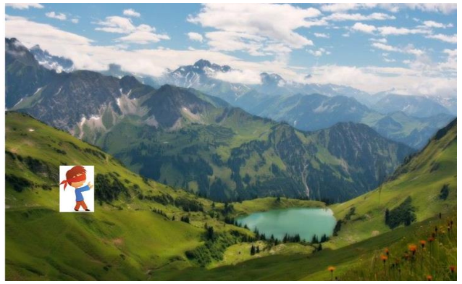
To start with

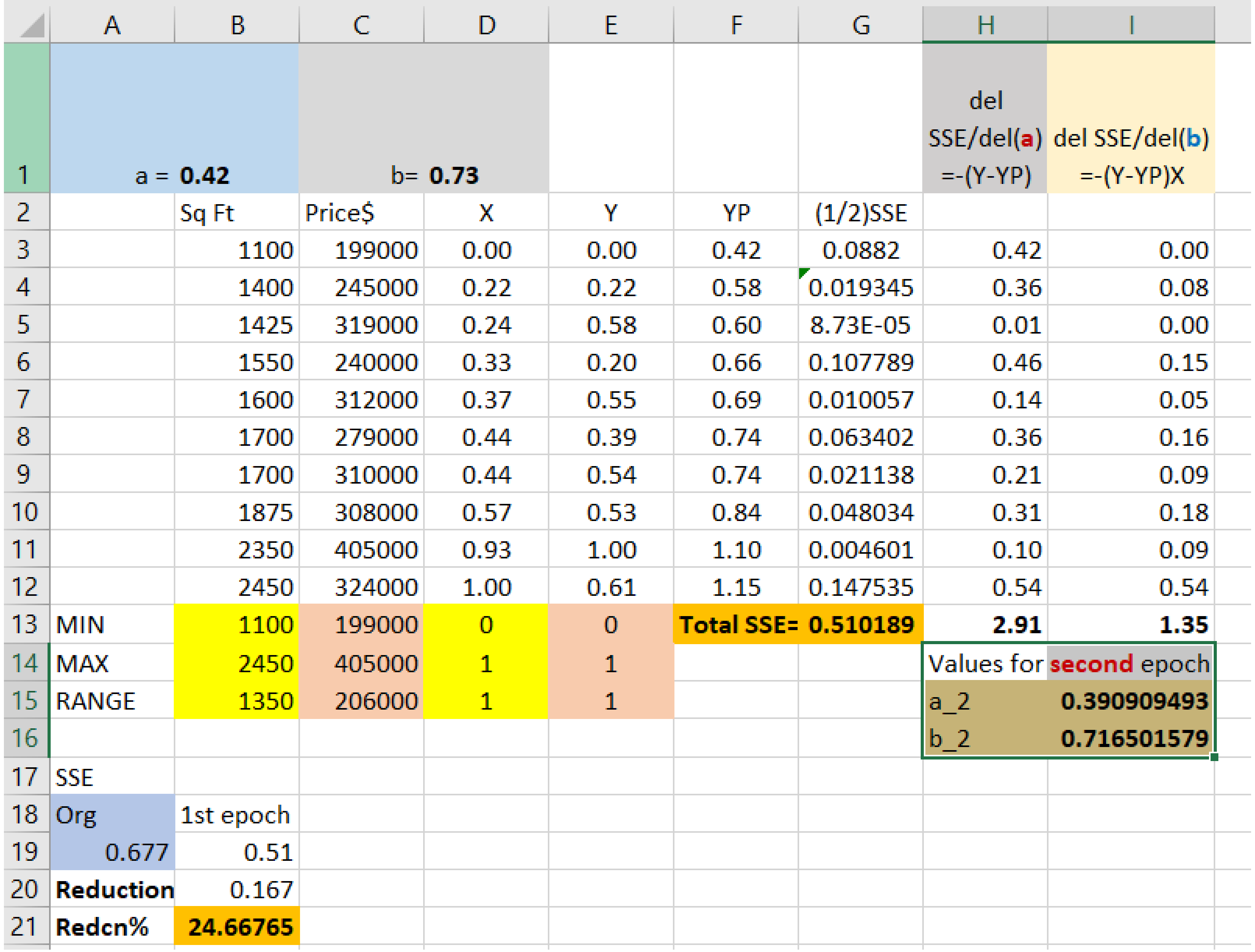




	A	B	C	D	E	F	G	H	I
1		a = **0.45**		b= **0.75**				del SSE/del(**a**) =-(Y-YP)	del SSE/del(**b**) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.45	0.10125	0.45	0.00
4		1400	245000	0.22	0.22	0.62	0.077368	0.39	0.09
5		1425	319000	0.24	0.58	0.63	0.001154	0.05	0.01
6		1550	240000	0.33	0.20	0.70	0.125486	0.50	0.17
7		1600	312000	0.37	0.55	0.73	0.016062	0.18	0.07
8		1700	279000	0.44	0.39	0.78	0.078006	0.39	0.18
9		1700	310000	0.44	0.54	0.78	0.02989	0.24	0.11
10		1875	308000	0.57	0.53	0.88	0.061751	0.35	0.20
11		2350	405000	0.93	1.00	1.14	0.010432	0.14	0.13
12		2450	324000	1.00	0.61	1.20	0.175945	0.59	0.59
13	MIN	1100	199000	0	0	**Total SSE=**	**0.677345**	**3.30**	**1.55**
14	MAX	2450	405000	1	1			Values for **first** epoch	
15	RANGE	1350	206000	1	1			a_1	**0.4169984**
16								b_1	**0.7345474**

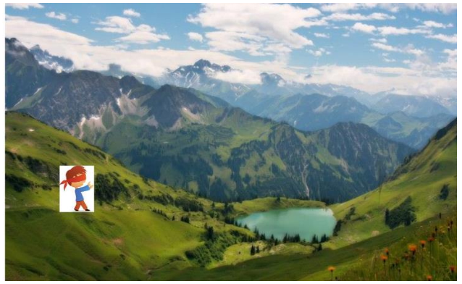
1st epoch

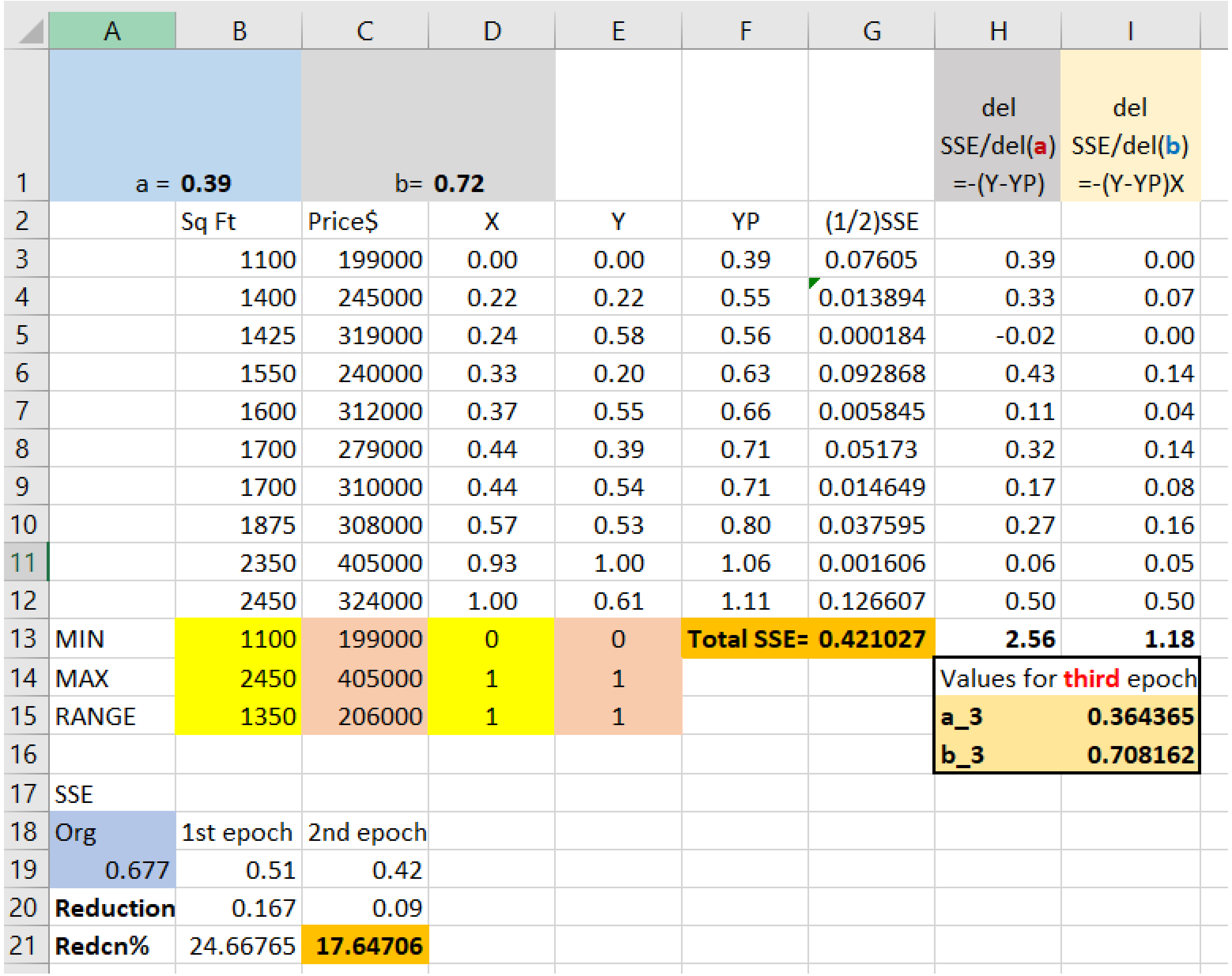




	A	B	C	D	E	F	G	H	I
1	a =	**0.42**	b=	**0.73**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.42	0.0882	0.42	0.00
4		1400	245000	0.22	0.22	0.58	0.019345	0.36	0.08
5		1425	319000	0.24	0.58	0.60	8.73E-05	0.01	0.00
6		1550	240000	0.33	0.20	0.66	0.107789	0.46	0.15
7		1600	312000	0.37	0.55	0.69	0.010057	0.14	0.05
8		1700	279000	0.44	0.39	0.74	0.063402	0.36	0.16
9		1700	310000	0.44	0.54	0.74	0.021138	0.21	0.09
10		1875	308000	0.57	0.53	0.84	0.048034	0.31	0.18
11		2350	405000	0.93	1.00	1.10	0.004601	0.10	0.09
12		2450	324000	1.00	0.61	1.15	0.147535	0.54	0.54
13	MIN	1100	199000	0	0	**Total SSE=**	**0.510189**	**2.91**	**1.35**
14	MAX	2450	405000	1	1			Values for second epoch	
15	RANGE	1350	206000	1	1			a_2	**0.390909493**
16								b_2	**0.716501579**
17	SSE								
18	Org	1st epoch							
19	0.677	0.51							
20	**Reduction**	0.167							
21	**Redcn%**	**24.66765**							

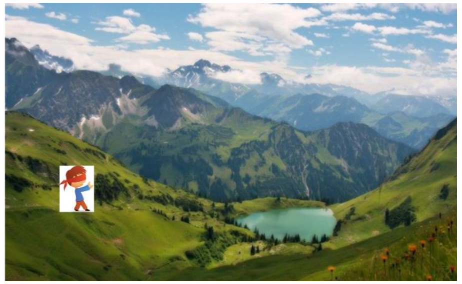
2nd epoch

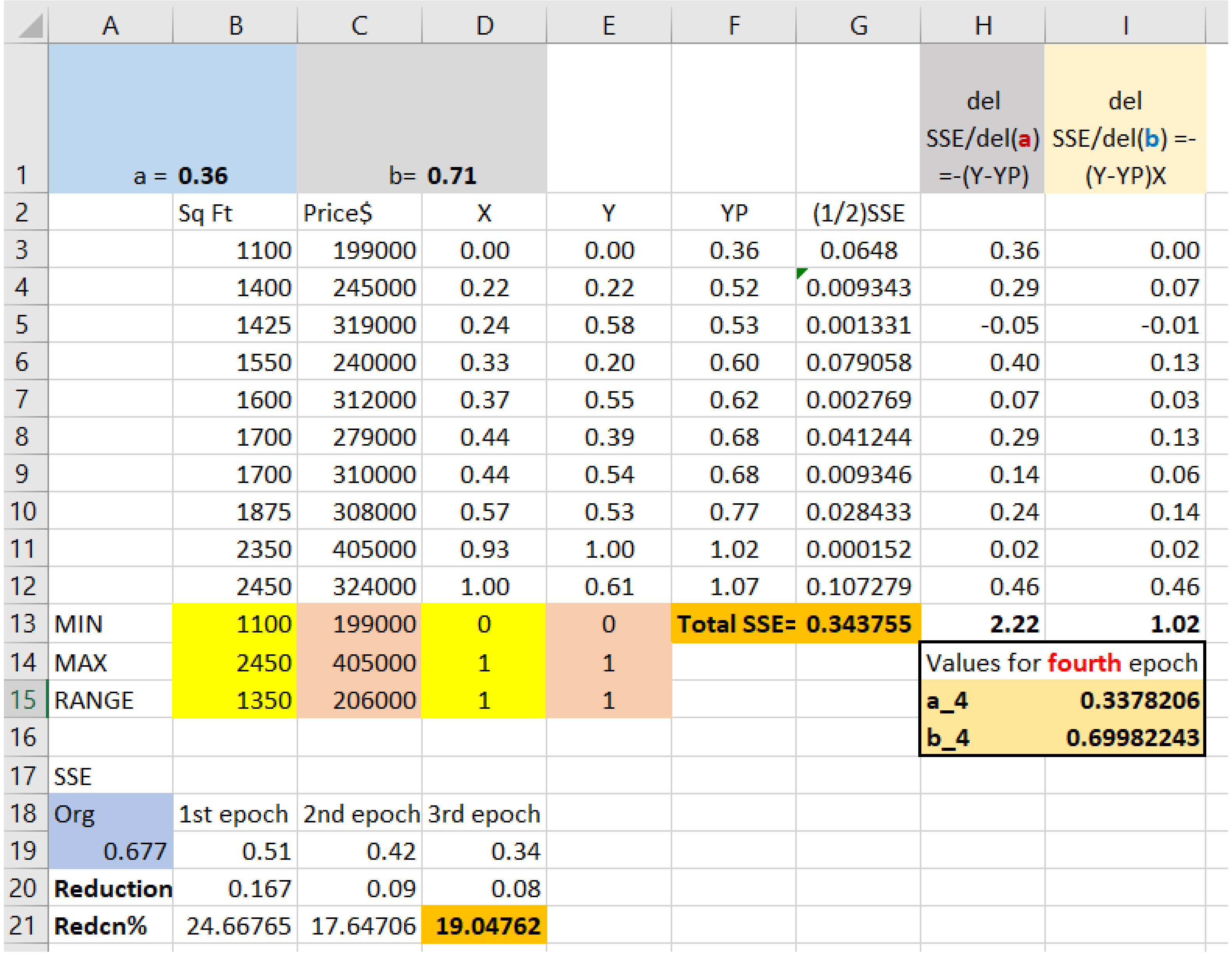




	A	B	C	D	E	F	G	H	I
1	a = **0.39**		b= **0.72**					del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.39	0.07605	0.39	0.00
4		1400	245000	0.22	0.22	0.55	0.013894	0.33	0.07
5		1425	319000	0.24	0.58	0.56	0.000184	-0.02	0.00
6		1550	240000	0.33	0.20	0.63	0.092868	0.43	0.14
7		1600	312000	0.37	0.55	0.66	0.005845	0.11	0.04
8		1700	279000	0.44	0.39	0.71	0.05173	0.32	0.14
9		1700	310000	0.44	0.54	0.71	0.014649	0.17	0.08
10		1875	308000	0.57	0.53	0.80	0.037595	0.27	0.16
11		2350	405000	0.93	1.00	1.06	0.001606	0.06	0.05
12		2450	324000	1.00	0.61	1.11	0.126607	0.50	0.50
13	MIN	1100	199000	0	0	**Total SSE=**	**0.421027**	**2.56**	**1.18**
14	MAX	2450	405000	1	1			Values for third epoch	
15	RANGE	1350	206000	1	1			**a_3**	**0.364365**
16								**b_3**	**0.708162**
17	SSE								
18	Org	1st epoch	2nd epoch						
19	0.677	0.51	0.42						
20	**Reduction**	0.167	0.09						
21	**Redcn%**	24.66765	**17.64706**						

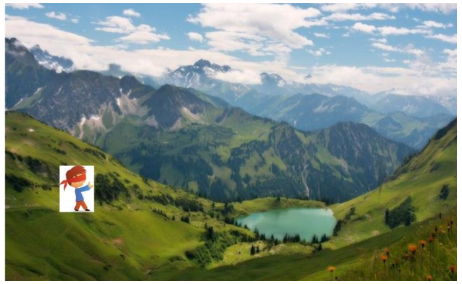
3rd epoch

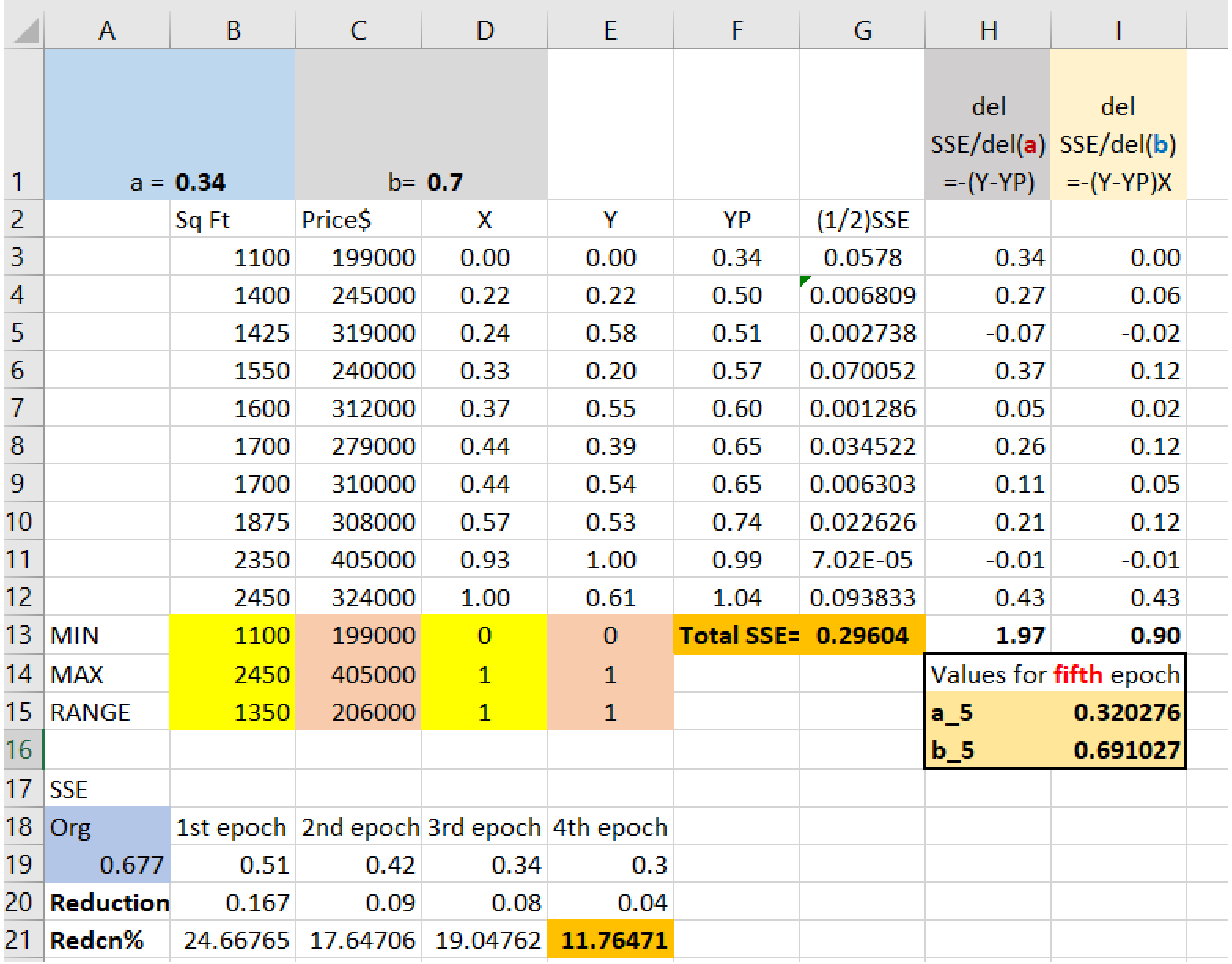




	A	B	C	D	E	F	G	H	I
1	a =	**0.36**	b=	**0.71**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.36	0.0648	0.36	0.00
4		1400	245000	0.22	0.22	0.52	0.009343	0.29	0.07
5		1425	319000	0.24	0.58	0.53	0.001331	-0.05	-0.01
6		1550	240000	0.33	0.20	0.60	0.079058	0.40	0.13
7		1600	312000	0.37	0.55	0.62	0.002769	0.07	0.03
8		1700	279000	0.44	0.39	0.68	0.041244	0.29	0.13
9		1700	310000	0.44	0.54	0.68	0.009346	0.14	0.06
10		1875	308000	0.57	0.53	0.77	0.028433	0.24	0.14
11		2350	405000	0.93	1.00	1.02	0.000152	0.02	0.02
12		2450	324000	1.00	0.61	1.07	0.107279	0.46	0.46
13	MIN	1100	199000	0	0	**Total SSE=**	**0.343755**	**2.22**	**1.02**
14	MAX	2450	405000	1	1			Values for fourth epoch	
15	RANGE	1350	206000	1	1			**a_4**	**0.3378206**
16								**b_4**	**0.69982243**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch					
19	0.677	0.51	0.42	0.34					
20	**Reduction**	0.167	0.09	0.08					
21	**Redcn%**	24.66765	17.64706	**19.04762**					

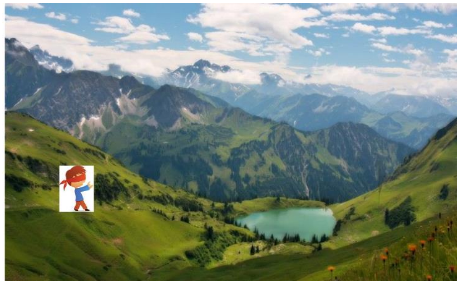
4th epoch

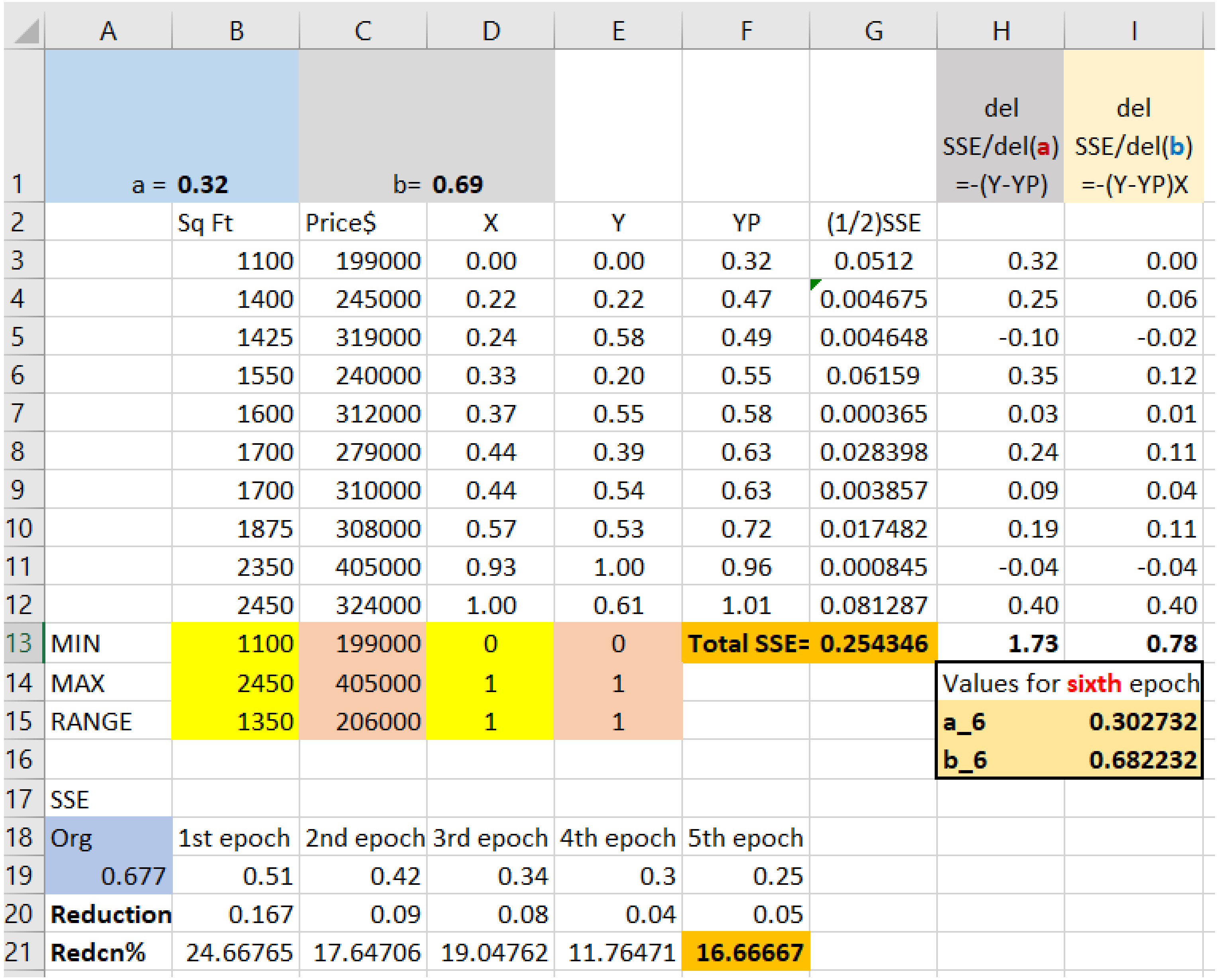




	A	B	C	D	E	F	G	H	I
1	a =	**0.34**	b=	**0.7**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.34	0.0578	0.34	0.00
4		1400	245000	0.22	0.22	0.50	0.006809	0.27	0.06
5		1425	319000	0.24	0.58	0.51	0.002738	-0.07	-0.02
6		1550	240000	0.33	0.20	0.57	0.070052	0.37	0.12
7		1600	312000	0.37	0.55	0.60	0.001286	0.05	0.02
8		1700	279000	0.44	0.39	0.65	0.034522	0.26	0.12
9		1700	310000	0.44	0.54	0.65	0.006303	0.11	0.05
10		1875	308000	0.57	0.53	0.74	0.022626	0.21	0.12
11		2350	405000	0.93	1.00	0.99	7.02E-05	-0.01	-0.01
12		2450	324000	1.00	0.61	1.04	0.093833	0.43	0.43
13	MIN	1100	199000	0	0	**Total SSE=**	**0.29604**	**1.97**	**0.90**
14	MAX	2450	405000	1	1			Values for **fifth** epoch	
15	RANGE	1350	206000	1	1			**a_5**	**0.320276**
16								**b_5**	**0.691027**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch				
19	0.677	0.51	0.42	0.34	0.3				
20	**Reduction**	0.167	0.09	0.08	0.04				
21	**Redcn%**	24.66765	17.64706	19.04762	**11.76471**				

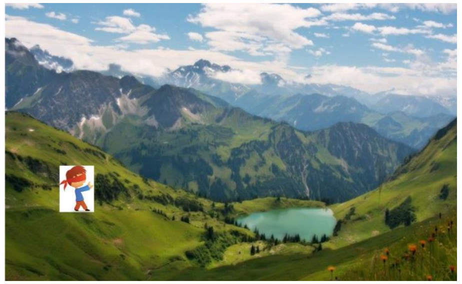
5th epoch

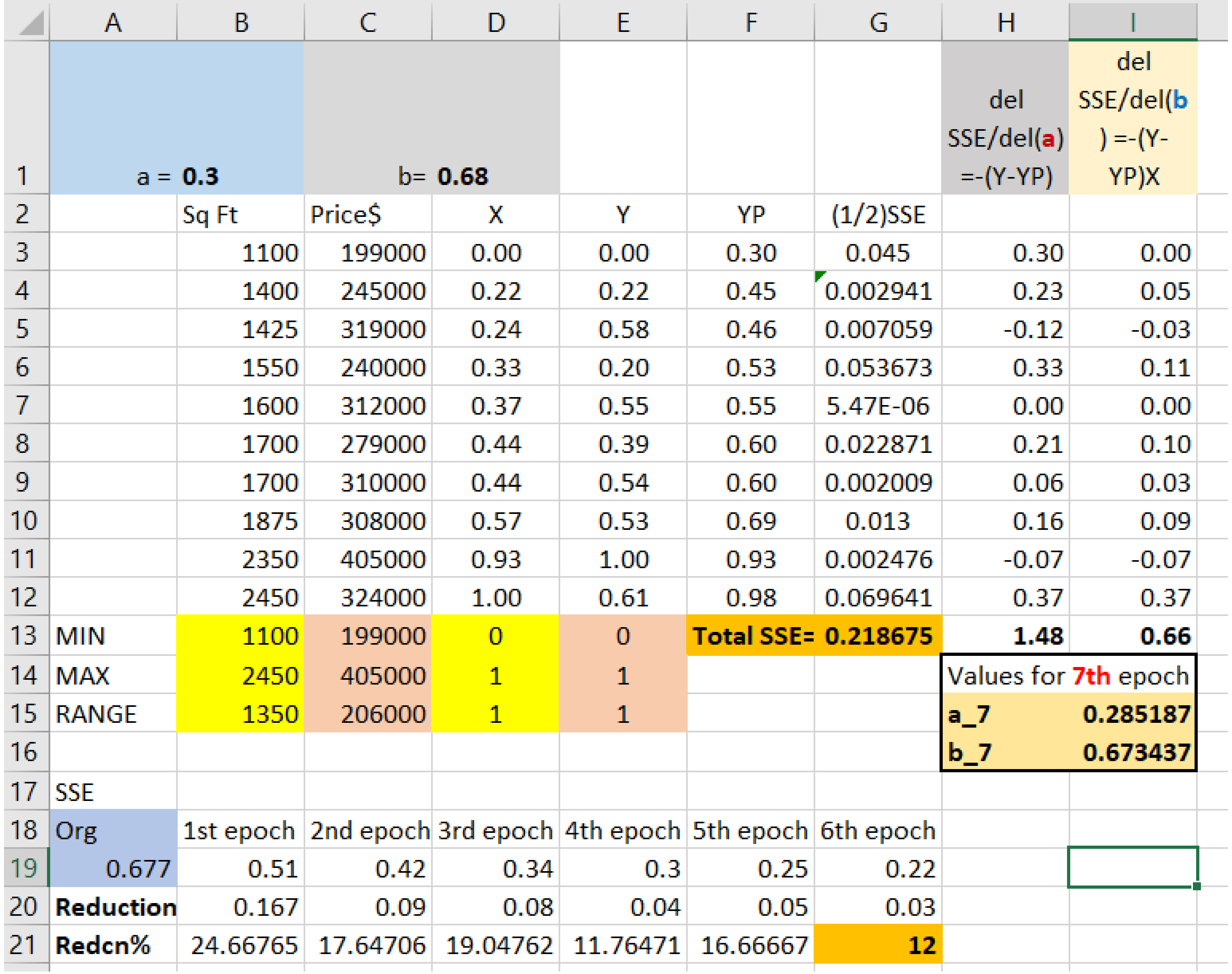




	A	B	C	D	E	F	G	H	I
1	a =	**0.32**	b=	**0.69**				del SSE/del(**a**) =-(Y-YP)	del SSE/del(**b**) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.32	0.0512	0.32	0.00
4		1400	245000	0.22	0.22	0.47	0.004675	0.25	0.06
5		1425	319000	0.24	0.58	0.49	0.004648	-0.10	-0.02
6		1550	240000	0.33	0.20	0.55	0.06159	0.35	0.12
7		1600	312000	0.37	0.55	0.58	0.000365	0.03	0.01
8		1700	279000	0.44	0.39	0.63	0.028398	0.24	0.11
9		1700	310000	0.44	0.54	0.63	0.003857	0.09	0.04
10		1875	308000	0.57	0.53	0.72	0.017482	0.19	0.11
11		2350	405000	0.93	1.00	0.96	0.000845	-0.04	-0.04
12		2450	324000	1.00	0.61	1.01	0.081287	0.40	0.40
13	MIN	1100	199000	0	0	**Total SSE=**	**0.254346**	**1.73**	**0.78**
14	MAX	2450	405000	1	1			Values for **sixth** epoch	
15	RANGE	1350	206000	1	1			**a_6**	**0.302732**
16								**b_6**	**0.682232**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch	5th epoch			
19	0.677	0.51	0.42	0.34	0.3	0.25			
20	**Reduction**	0.167	0.09	0.08	0.04	0.05			
21	**Redcn%**	24.66765	17.64706	19.04762	11.76471	**16.66667**			

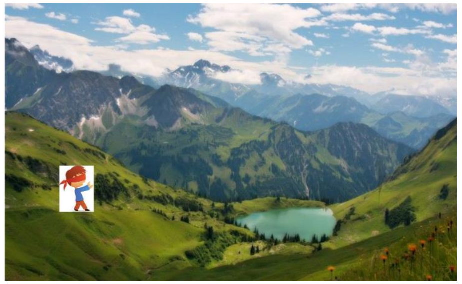
6th epoch

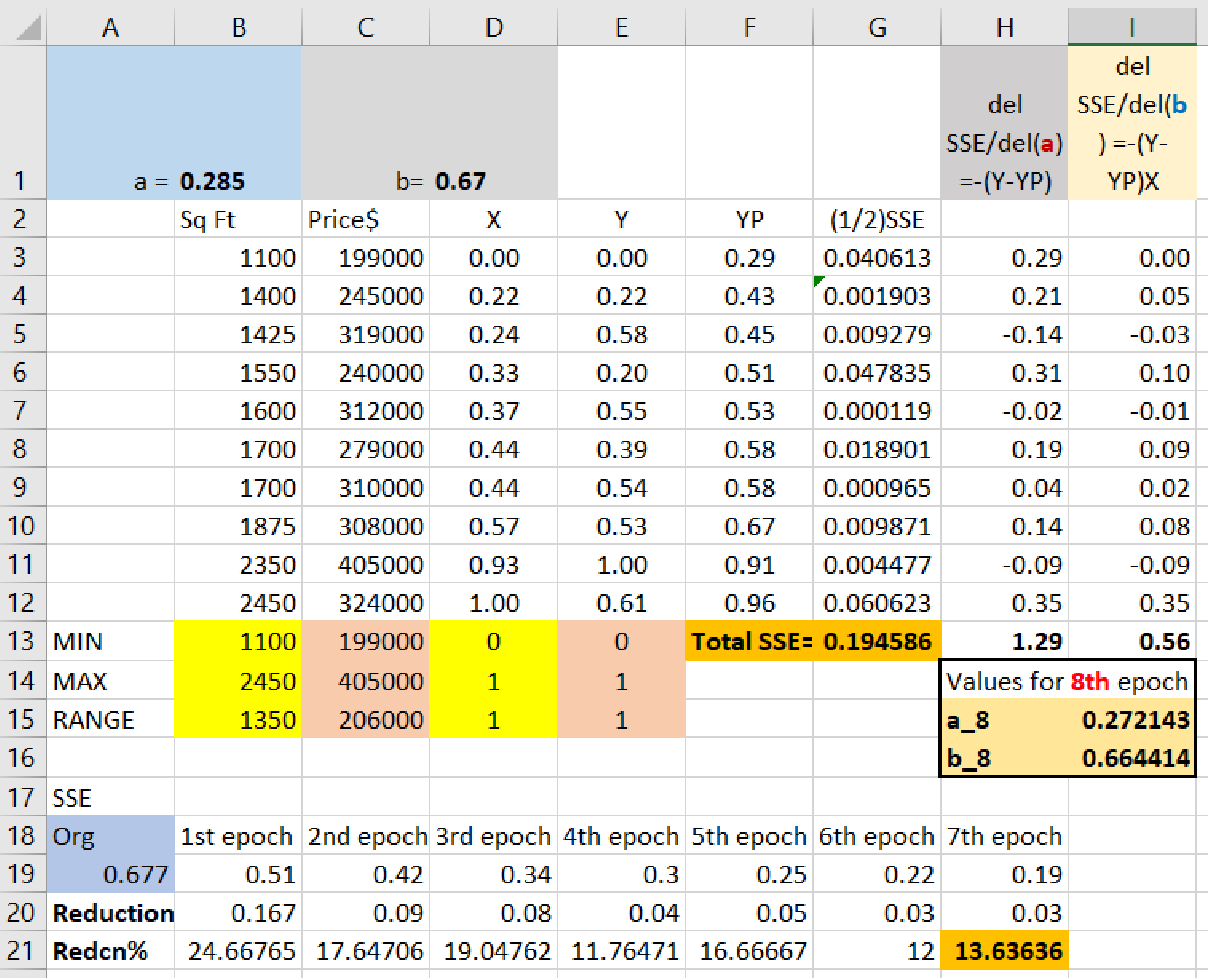




	A	B	C	D	E	F	G	H	I
1	a =	**0.3**	b=	**0.68**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.30	0.045	0.30	0.00
4		1400	245000	0.22	0.22	0.45	0.002941	0.23	0.05
5		1425	319000	0.24	0.58	0.46	0.007059	-0.12	-0.03
6		1550	240000	0.33	0.20	0.53	0.053673	0.33	0.11
7		1600	312000	0.37	0.55	0.55	5.47E-06	0.00	0.00
8		1700	279000	0.44	0.39	0.60	0.022871	0.21	0.10
9		1700	310000	0.44	0.54	0.60	0.002009	0.06	0.03
10		1875	308000	0.57	0.53	0.69	0.013	0.16	0.09
11		2350	405000	0.93	1.00	0.93	0.002476	-0.07	-0.07
12		2450	324000	1.00	0.61	0.98	0.069641	0.37	0.37
13	MIN	1100	199000	0	0	**Total SSE=**	**0.218675**	**1.48**	**0.66**
14	MAX	2450	405000	1	1			Values for 7th epoch	
15	RANGE	1350	206000	1	1			**a_7**	**0.285187**
16								**b_7**	**0.673437**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch	5th epoch	6th epoch		
19	0.677	0.51	0.42	0.34	0.3	0.25	0.22		
20	**Reduction**	0.167	0.09	0.08	0.04	0.05	0.03		
21	**Redcn%**	24.66765	17.64706	19.04762	11.76471	16.66667	**12**		

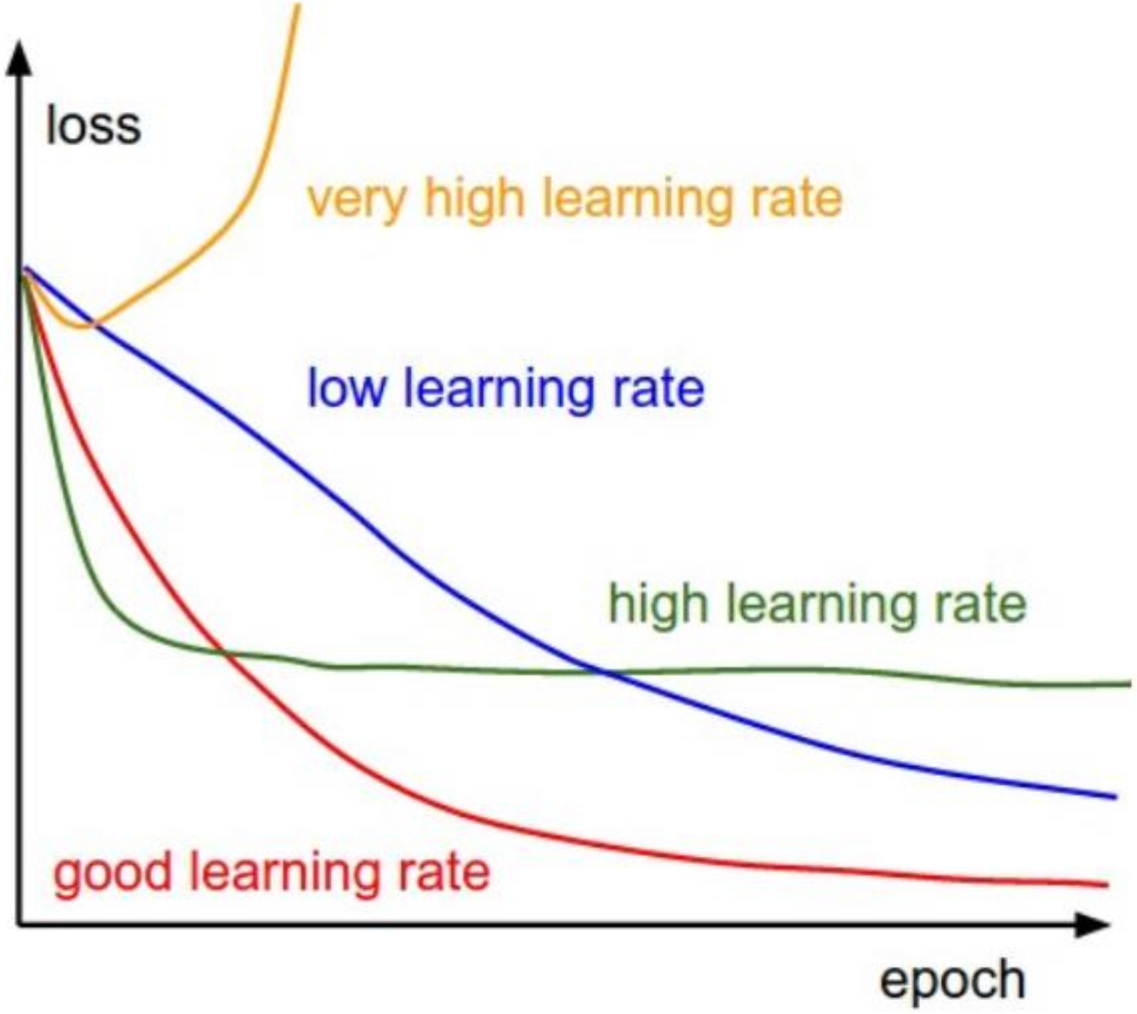
7th epoch

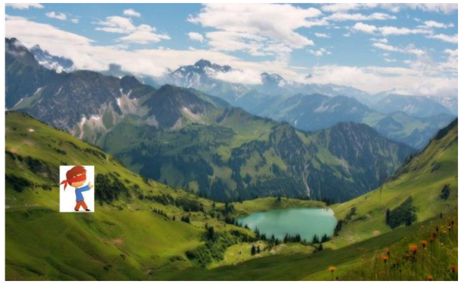




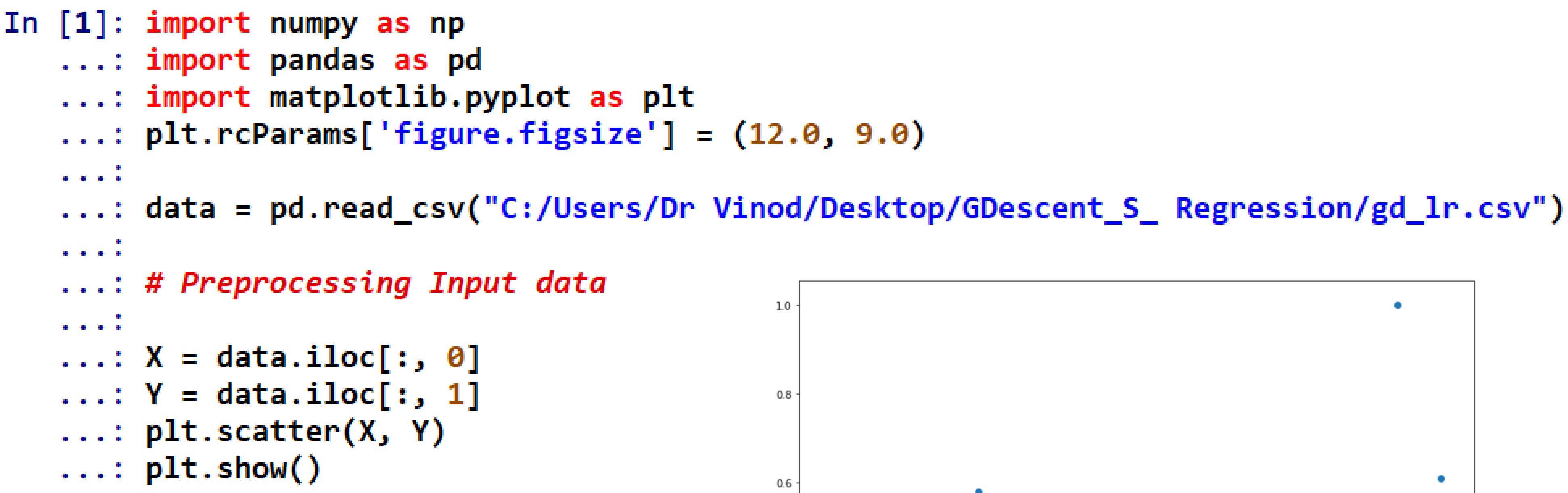
	A	B	C	D	E	F	G	H	I
1	a =	**0.285**	b=	**0.67**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.29	0.040613	0.29	0.00
4		1400	245000	0.22	0.22	0.43	0.001903	0.21	0.05
5		1425	319000	0.24	0.58	0.45	0.009279	-0.14	-0.03
6		1550	240000	0.33	0.20	0.51	0.047835	0.31	0.10
7		1600	312000	0.37	0.55	0.53	0.000119	-0.02	-0.01
8		1700	279000	0.44	0.39	0.58	0.018901	0.19	0.09
9		1700	310000	0.44	0.54	0.58	0.000965	0.04	0.02
10		1875	308000	0.57	0.53	0.67	0.009871	0.14	0.08
11		2350	405000	0.93	1.00	0.91	0.004477	-0.09	-0.09
12		2450	324000	1.00	0.61	0.96	0.060623	0.35	0.35
13	MIN	1100	199000	0	0	**Total SSE=**	**0.194586**	**1.29**	**0.56**
14	MAX	2450	405000	1	1			Values for 8th epoch	
15	RANGE	1350	206000	1	1			**a_8**	**0.272143**
16								**b_8**	**0.664414**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch	5th epoch	6th epoch	7th epoch	
19	0.677	0.51	0.42	0.34	0.3	0.25	0.22	0.19	
20	**Reduction**	0.167	0.09	0.08	0.04	0.05	0.03	0.03	
21	**Redcn%**	24.66765	17.64706	19.04762	11.76471	16.66667	12	**13.63636**	

Learning Rate vs Error

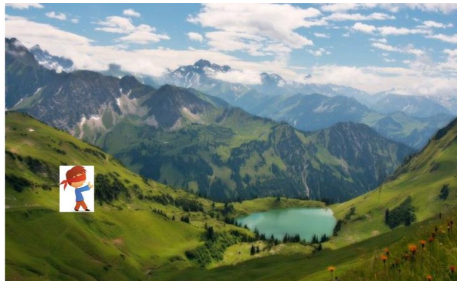


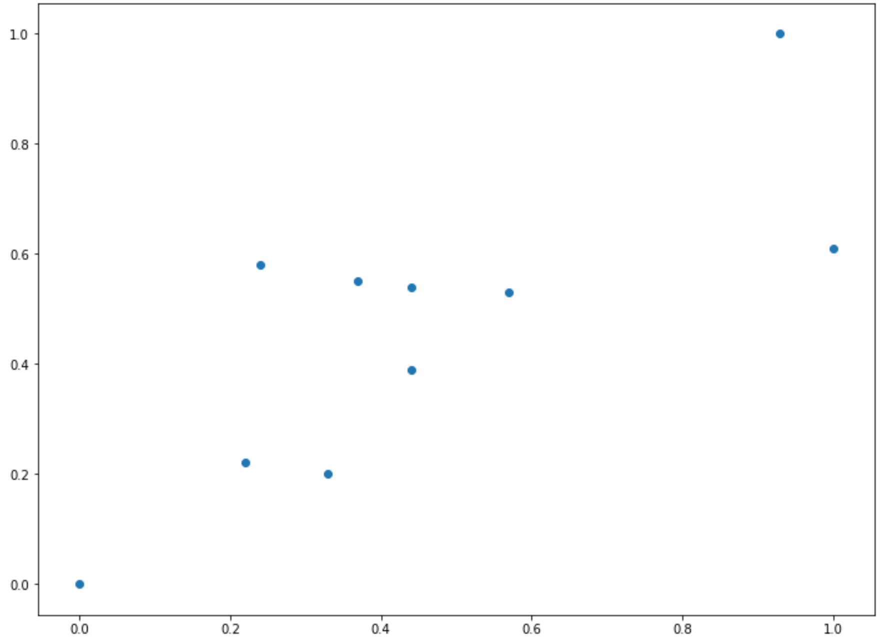


Lets do it in Python

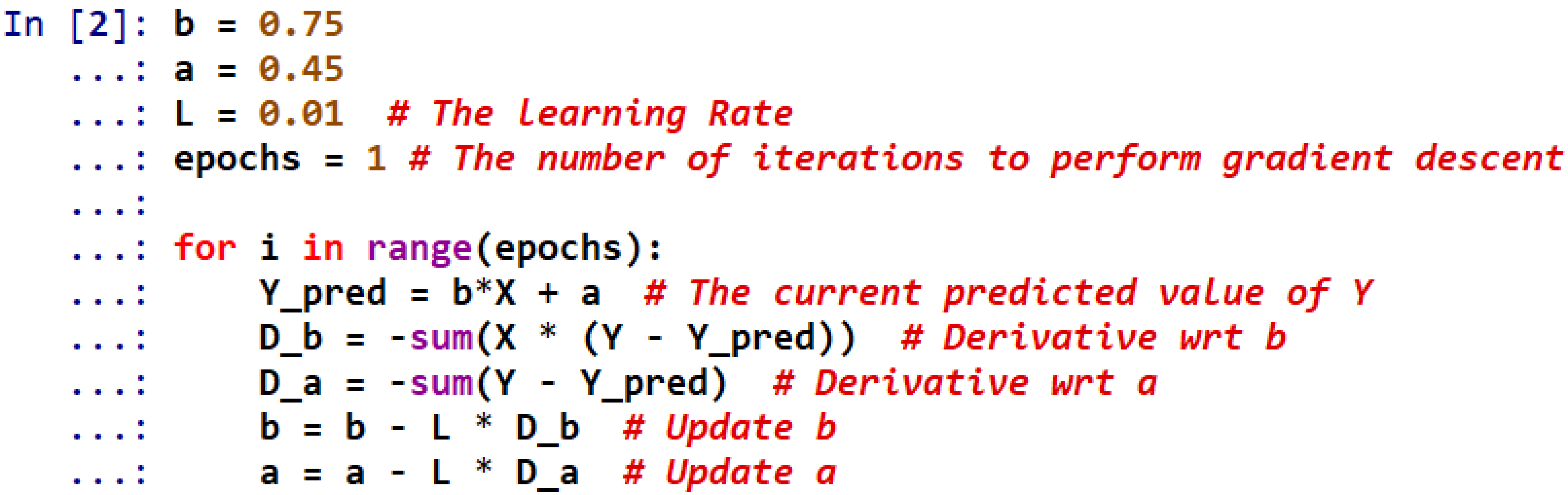


```
In [1]: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: plt.rcParams['figure.figsize'] = (12.0, 9.0)
   ...:
   ...: data = pd.read_csv("C:/Users/Dr Vinod/Desktop/GDescent_S_ Regression/gd_lr.csv")
   ...:
   ...: # Preprocessing Input data
   ...:
   ...: X = data.iloc[:, 0]
   ...: Y = data.iloc[:, 1]
   ...: plt.scatter(X, Y)
   ...: plt.show()
```

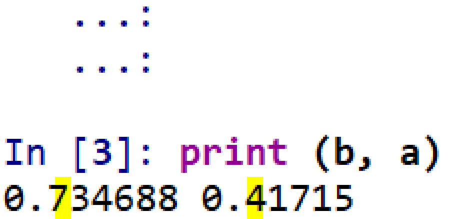




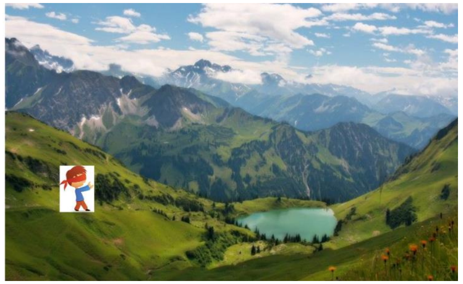
1^{st} Epoch



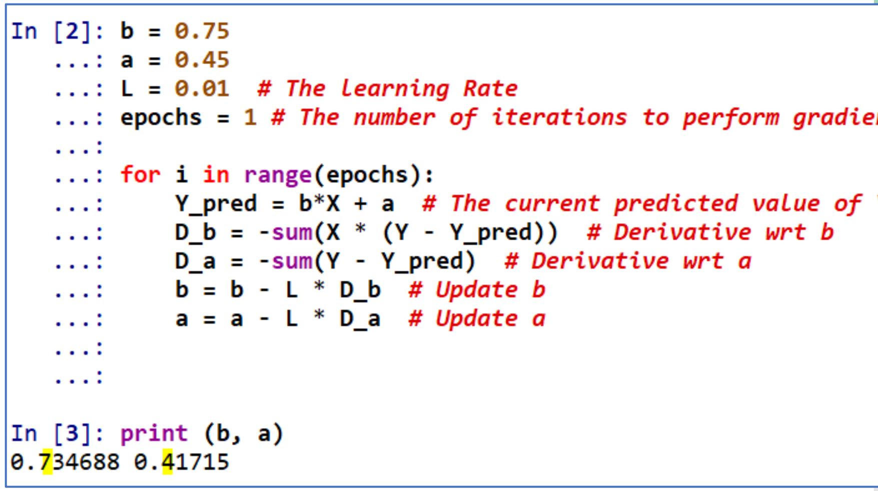
```
In [2]: b = 0.75
   ...: a = 0.45
   ...: L = 0.01  # The learning Rate
   ...: epochs = 1 # The number of iterations to perform gradient descent
   ...:
   ...: for i in range(epochs):
   ...:     Y_pred = b*X + a  # The current predicted value of Y
   ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
   ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
   ...:     b = b - L * D_b  # Update b
   ...:     a = a - L * D_a  # Update a
```



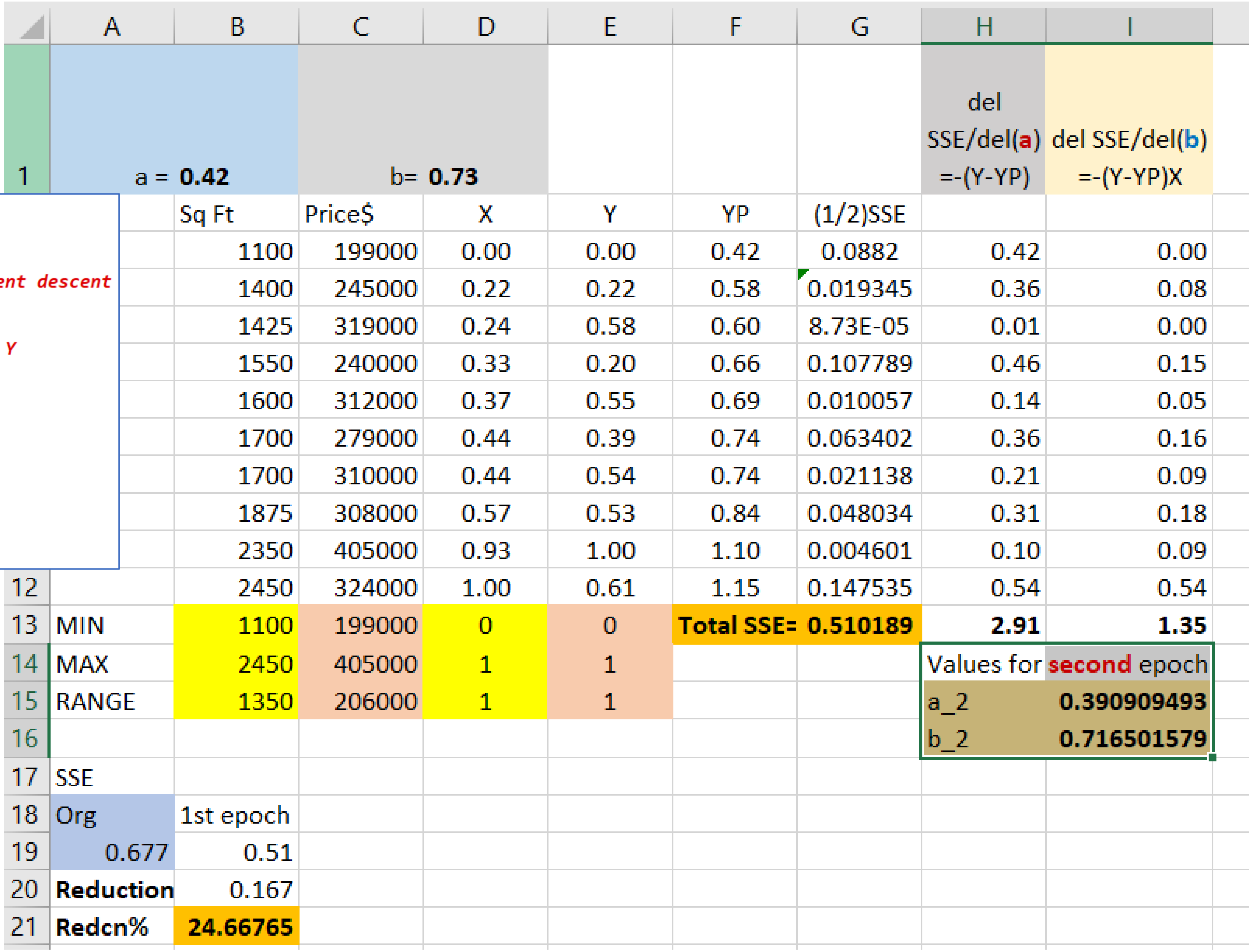
```
   ...:
   ...:

In [3]: print (b, a)
0.734688 0.41715
```



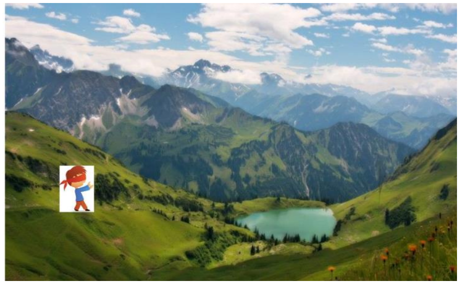
1st epoch



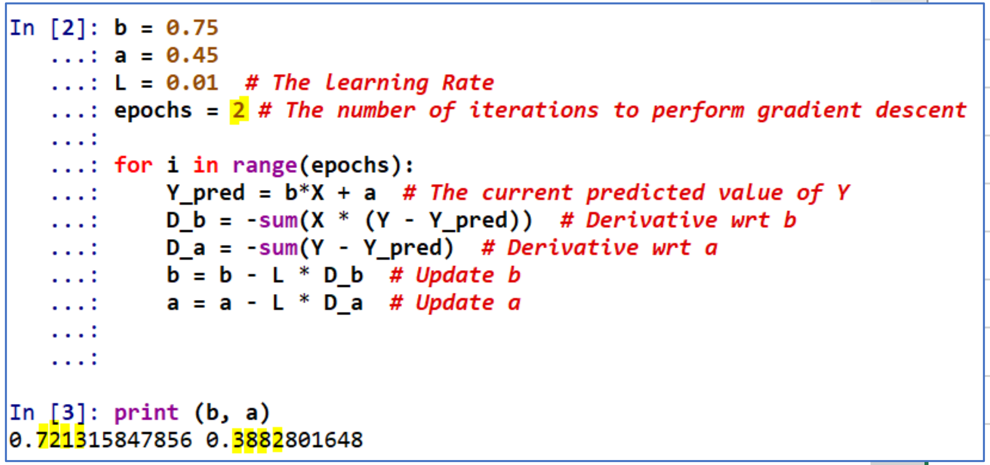
```
In [2]: b = 0.75
   ...: a = 0.45
   ...: L = 0.01  # The learning Rate
   ...: epochs = 1 # The number of iterations to perform gradient descent
   ...:
   ...: for i in range(epochs):
   ...:     Y_pred = b*X + a  # The current predicted value of Y
   ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
   ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
   ...:     b = b - L * D_b  # Update b
   ...:     a = a - L * D_a  # Update a
   ...:
   ...:

In [3]: print (b, a)
0.734688 0.41715
```



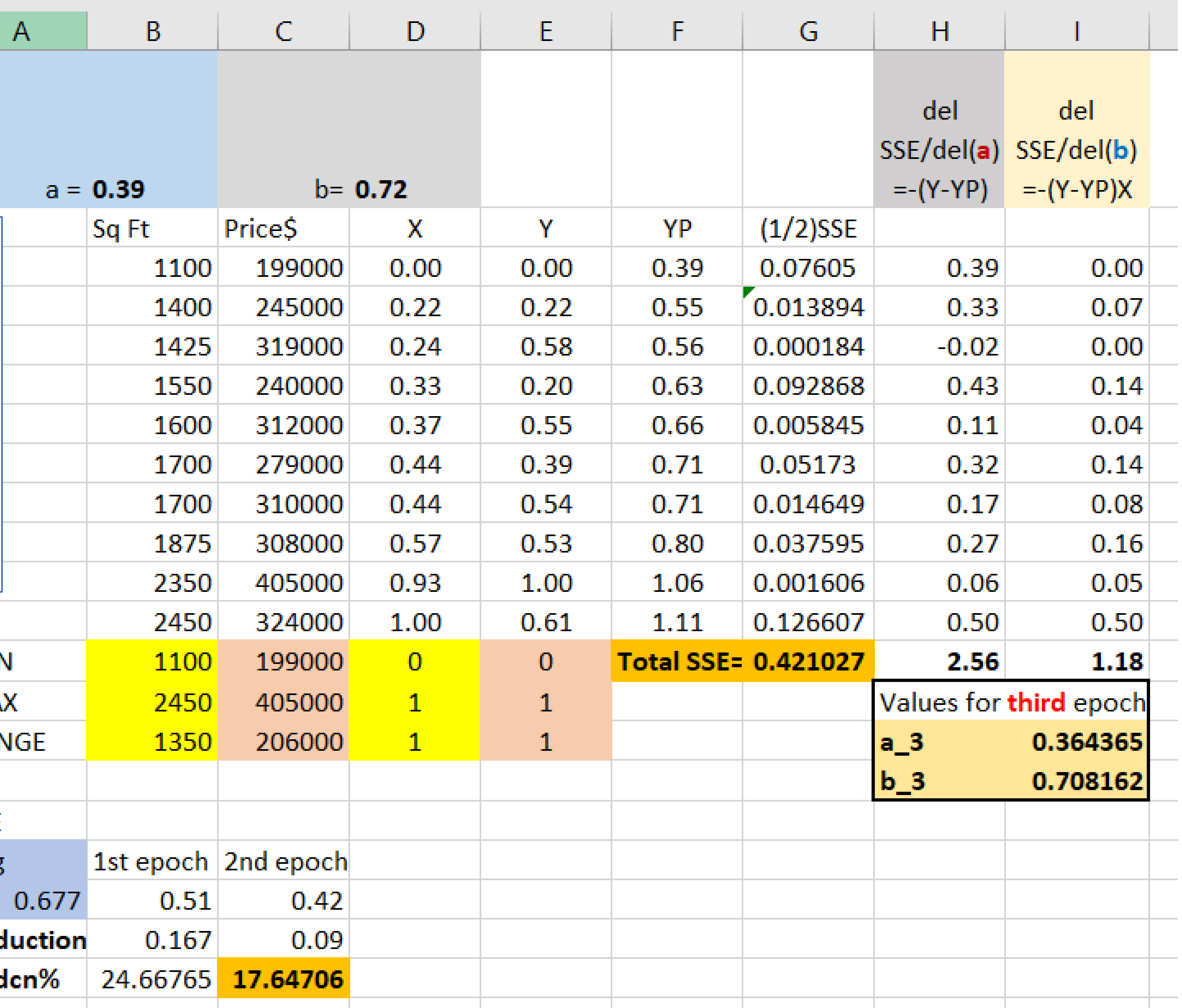
	A	B	C	D	E	F	G	H	I
1	a =	**0.42**	b=	**0.73**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.42	0.0882	0.42	0.00
		1400	245000	0.22	0.22	0.58	0.019345	0.36	0.08
		1425	319000	0.24	0.58	0.60	8.73E-05	0.01	0.00
		1550	240000	0.33	0.20	0.66	0.107789	0.46	0.15
		1600	312000	0.37	0.55	0.69	0.010057	0.14	0.05
		1700	279000	0.44	0.39	0.74	0.063402	0.36	0.16
		1700	310000	0.44	0.54	0.74	0.021138	0.21	0.09
		1875	308000	0.57	0.53	0.84	0.048034	0.31	0.18
		2350	405000	0.93	1.00	1.10	0.004601	0.10	0.09
12		2450	324000	1.00	0.61	1.15	0.147535	0.54	0.54
13	MIN	1100	199000	0	0	**Total SSE=**	**0.510189**	**2.91**	**1.35**
14	MAX	2450	405000	1	1			Values for **second** epoch	
15	RANGE	1350	206000	1	1			a_2	**0.390909493**
16								b_2	**0.716501579**
17	SSE								
18	Org	1st epoch							
19	0.677	0.51							
20	**Reduction**	0.167							
21	**Redcn%**	**24.66765**							



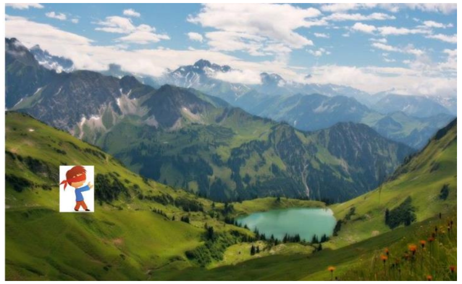
2nd epoch



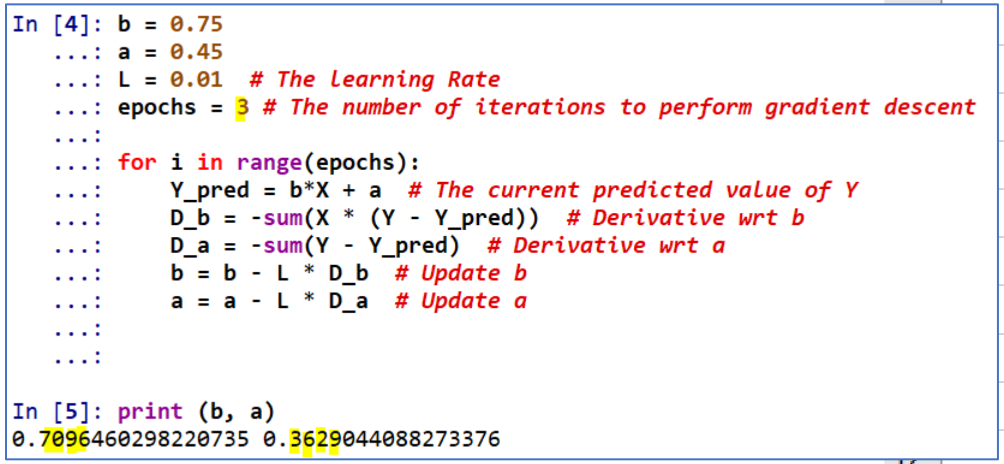
```
In [2]: b = 0.75
   ...: a = 0.45
   ...: L = 0.01  # The learning Rate
   ...: epochs = 2 # The number of iterations to perform gradient descent
   ...:
   ...: for i in range(epochs):
   ...:     Y_pred = b*X + a  # The current predicted value of Y
   ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
   ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
   ...:     b = b - L * D_b  # Update b
   ...:     a = a - L * D_a  # Update a
   ...:
   ...:

In [3]: print (b, a)
0.721315847856 0.3882801648
```



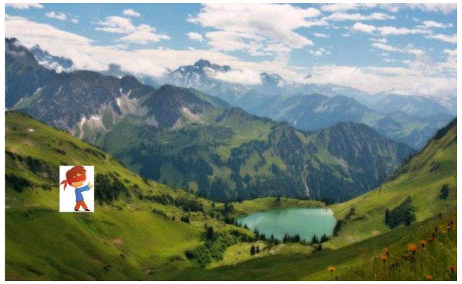
	A	B	C	D	E	F	G	H	I
1	a = **0.39**		b= **0.72**					del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.39	0.07605	0.39	0.00
		1400	245000	0.22	0.22	0.55	0.013894	0.33	0.07
		1425	319000	0.24	0.58	0.56	0.000184	-0.02	0.00
		1550	240000	0.33	0.20	0.63	0.092868	0.43	0.14
		1600	312000	0.37	0.55	0.66	0.005845	0.11	0.04
		1700	279000	0.44	0.39	0.71	0.05173	0.32	0.14
		1700	310000	0.44	0.54	0.71	0.014649	0.17	0.08
		1875	308000	0.57	0.53	0.80	0.037595	0.27	0.16
		2350	405000	0.93	1.00	1.06	0.001606	0.06	0.05
12		2450	324000	1.00	0.61	1.11	0.126607	0.50	0.50
13	MIN	1100	199000	0	0	**Total SSE=**	**0.421027**	**2.56**	**1.18**
14	MAX	2450	405000	1	1			Values for **third** epoch	
15	RANGE	1350	206000	1	1			**a_3**	**0.364365**
16								**b_3**	**0.708162**
17	SSE								
18	Org	1st epoch	2nd epoch						
19	0.677	0.51	0.42						
20	**Reduction**	0.167	0.09						
21	**Redcn%**	24.66765	**17.64706**						

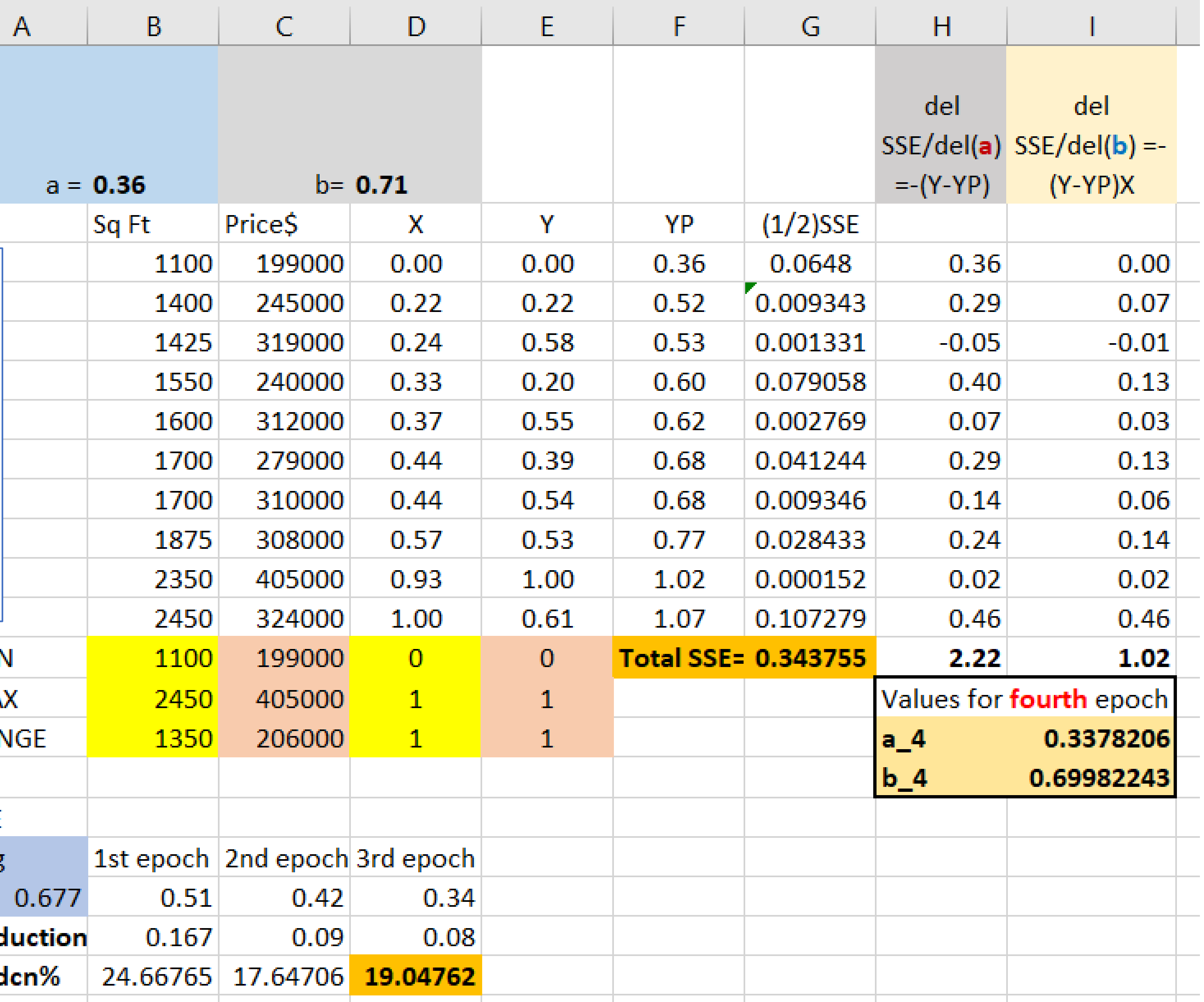


3rd epoch



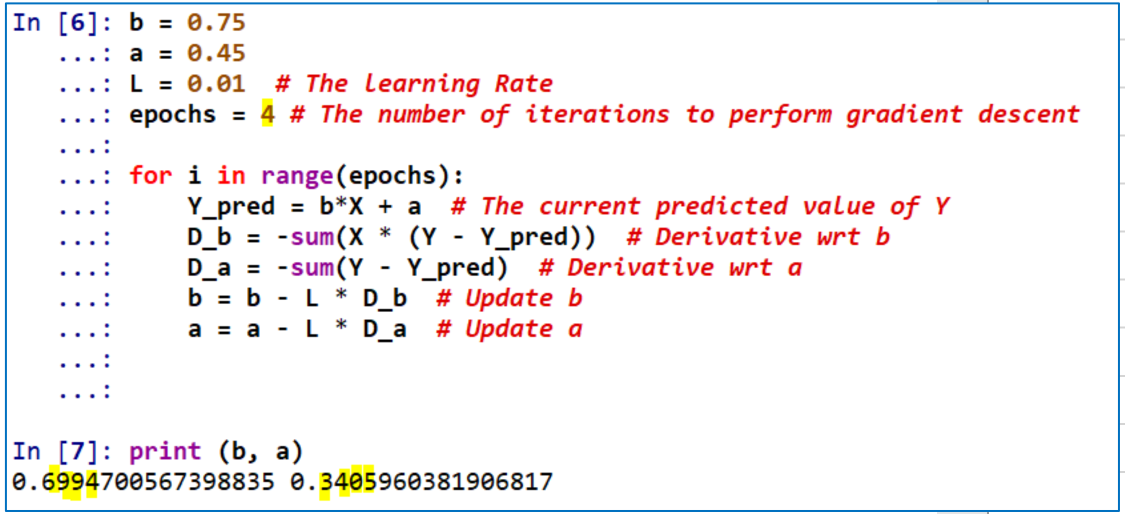
```
In [4]: b = 0.75
   ...: a = 0.45
   ...: L = 0.01  # The learning Rate
   ...: epochs = 3 # The number of iterations to perform gradient descent
   ...:
   ...: for i in range(epochs):
   ...:     Y_pred = b*X + a  # The current predicted value of Y
   ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
   ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
   ...:     b = b - L * D_b  # Update b
   ...:     a = a - L * D_a  # Update a
   ...:
   ...:

In [5]: print (b, a)
0.7096460298220735 0.3629044088273376
```



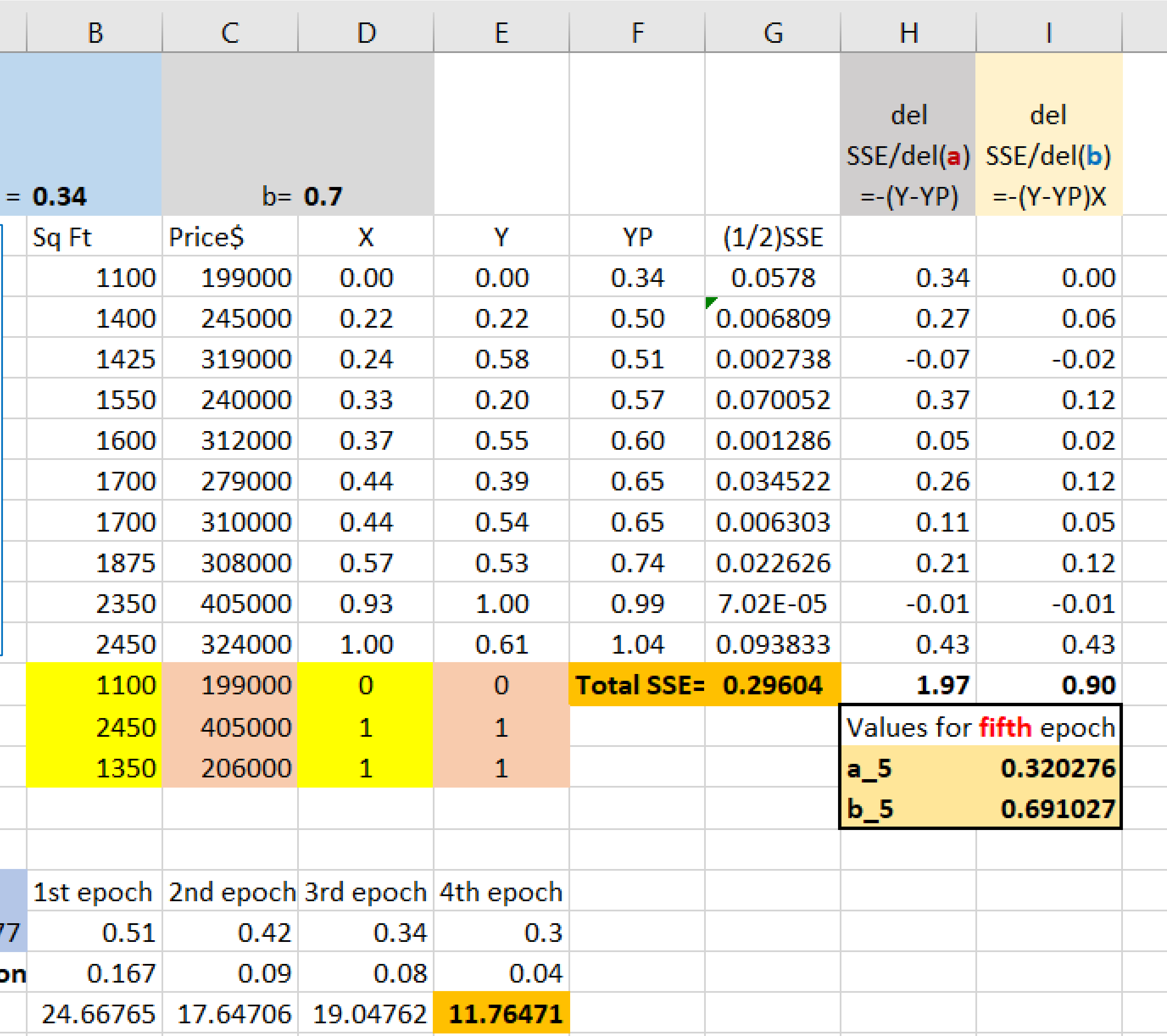


	A	B	C	D	E	F	G	H	I
1	a = **0.36**		b= **0.71**					del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.36	0.0648	0.36	0.00
		1400	245000	0.22	0.22	0.52	0.009343	0.29	0.07
		1425	319000	0.24	0.58	0.53	0.001331	-0.05	-0.01
		1550	240000	0.33	0.20	0.60	0.079058	0.40	0.13
		1600	312000	0.37	0.55	0.62	0.002769	0.07	0.03
		1700	279000	0.44	0.39	0.68	0.041244	0.29	0.13
		1700	310000	0.44	0.54	0.68	0.009346	0.14	0.06
		1875	308000	0.57	0.53	0.77	0.028433	0.24	0.14
		2350	405000	0.93	1.00	1.02	0.000152	0.02	0.02
		2450	324000	1.00	0.61	1.07	0.107279	0.46	0.46
13	MIN	1100	199000	0	0	**Total SSE=**	**0.343755**	**2.22**	**1.02**
14	MAX	2450	405000	1	1			Values for fourth epoch	
15	RANGE	1350	206000	1	1			**a_4**	**0.3378206**
16								**b_4**	**0.69982243**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch					
19	0.677	0.51	0.42	0.34					
20	**Reduction**	0.167	0.09	0.08					
21	**Redcn%**	24.66765	17.64706	**19.04762**					

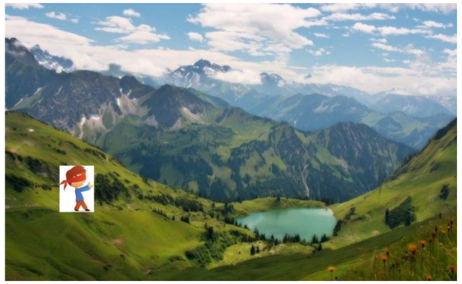
4th epoch



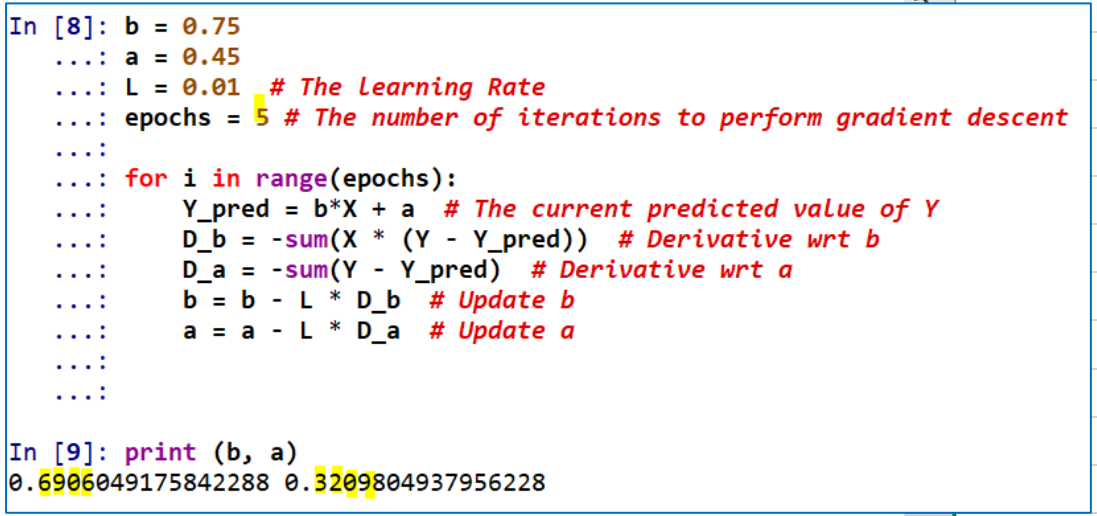
```
In [6]: b = 0.75
   ...: a = 0.45
   ...: L = 0.01  # The learning Rate
   ...: epochs = 4 # The number of iterations to perform gradient descent
   ...:
   ...: for i in range(epochs):
   ...:     Y_pred = b*X + a  # The current predicted value of Y
   ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
   ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
   ...:     b = b - L * D_b  # Update b
   ...:     a = a - L * D_a  # Update a
   ...:
   ...:

In [7]: print (b, a)
0.6994700567398835 0.3405960381906817
```



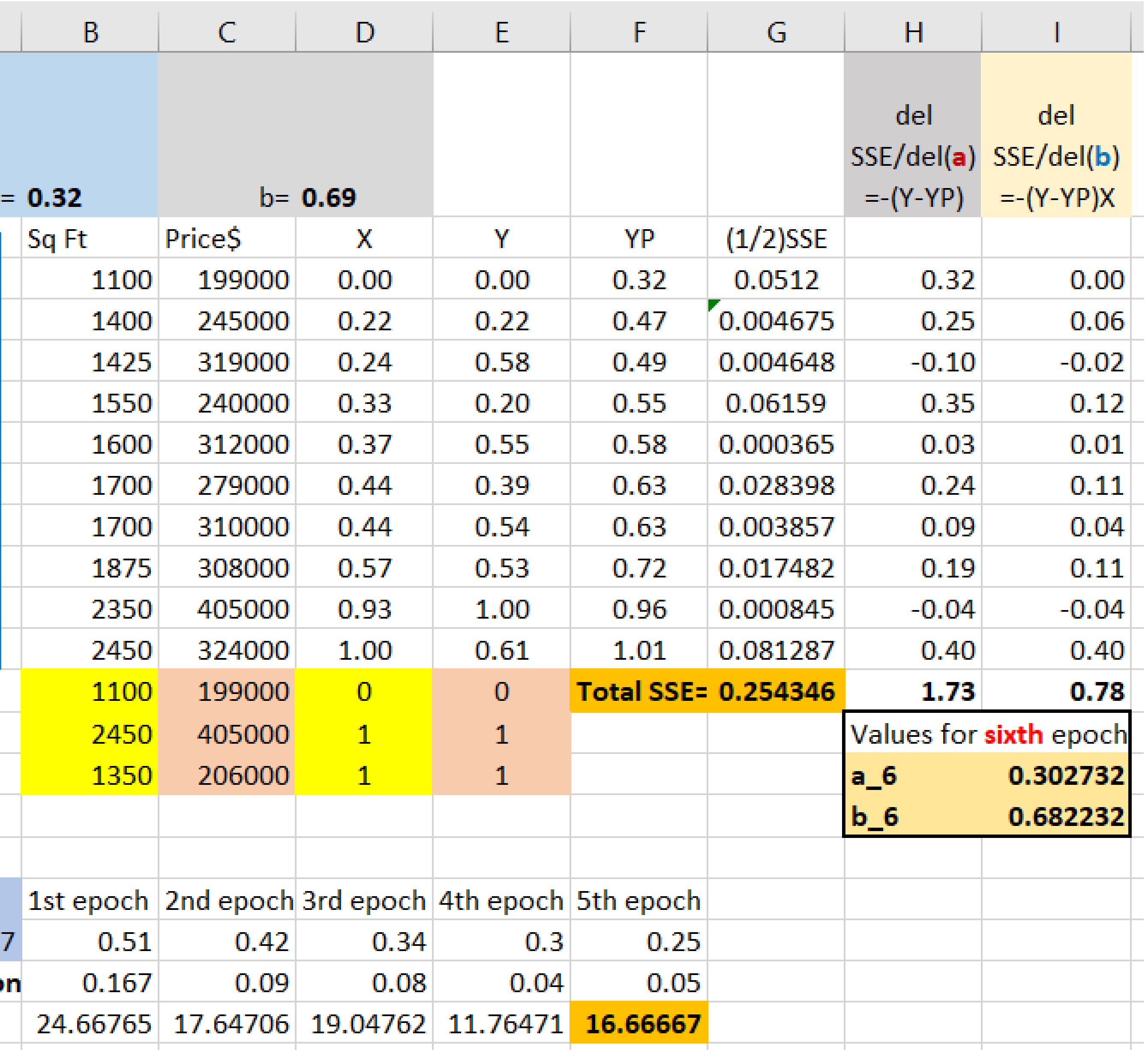
	A	B	C	D	E	F	G	H	I
1	a = **0.34**		b= **0.7**					del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.34	0.0578	0.34	0.00
		1400	245000	0.22	0.22	0.50	0.006809	0.27	0.06
		1425	319000	0.24	0.58	0.51	0.002738	-0.07	-0.02
		1550	240000	0.33	0.20	0.57	0.070052	0.37	0.12
		1600	312000	0.37	0.55	0.60	0.001286	0.05	0.02
		1700	279000	0.44	0.39	0.65	0.034522	0.26	0.12
		1700	310000	0.44	0.54	0.65	0.006303	0.11	0.05
		1875	308000	0.57	0.53	0.74	0.022626	0.21	0.12
		2350	405000	0.93	1.00	0.99	7.02E-05	-0.01	-0.01
		2450	324000	1.00	0.61	1.04	0.093833	0.43	0.43
13	MIN	1100	199000	0	0	**Total SSE=**	**0.29604**	**1.97**	**0.90**
14	MAX	2450	405000	1	1			Values for fifth epoch	
15	RANGE	1350	206000	1	1			**a_5**	**0.320276**
16								**b_5**	**0.691027**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch				
19	0.677	0.51	0.42	0.34	0.3				
20	**Reduction**	0.167	0.09	0.08	0.04				
21	**Redcn%**	24.66765	17.64706	19.04762	**11.76471**				



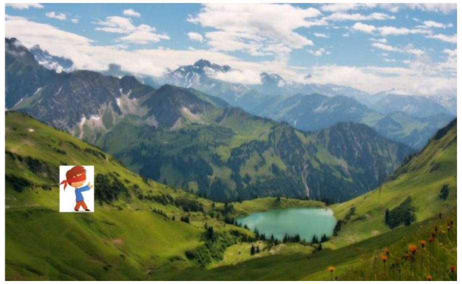
5th epoch



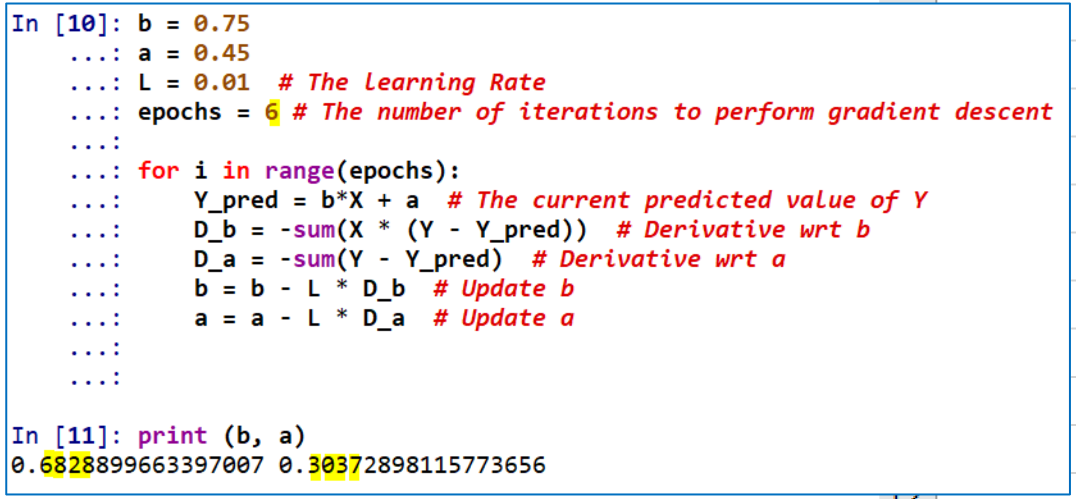
```
In [8]: b = 0.75
   ...: a = 0.45
   ...: L = 0.01  # The learning Rate
   ...: epochs = 5 # The number of iterations to perform gradient descent
   ...:
   ...: for i in range(epochs):
   ...:     Y_pred = b*X + a  # The current predicted value of Y
   ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
   ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
   ...:     b = b - L * D_b  # Update b
   ...:     a = a - L * D_a  # Update a
   ...:
   ...:

In [9]: print (b, a)
0.6906049175842288 0.3209804937956228
```



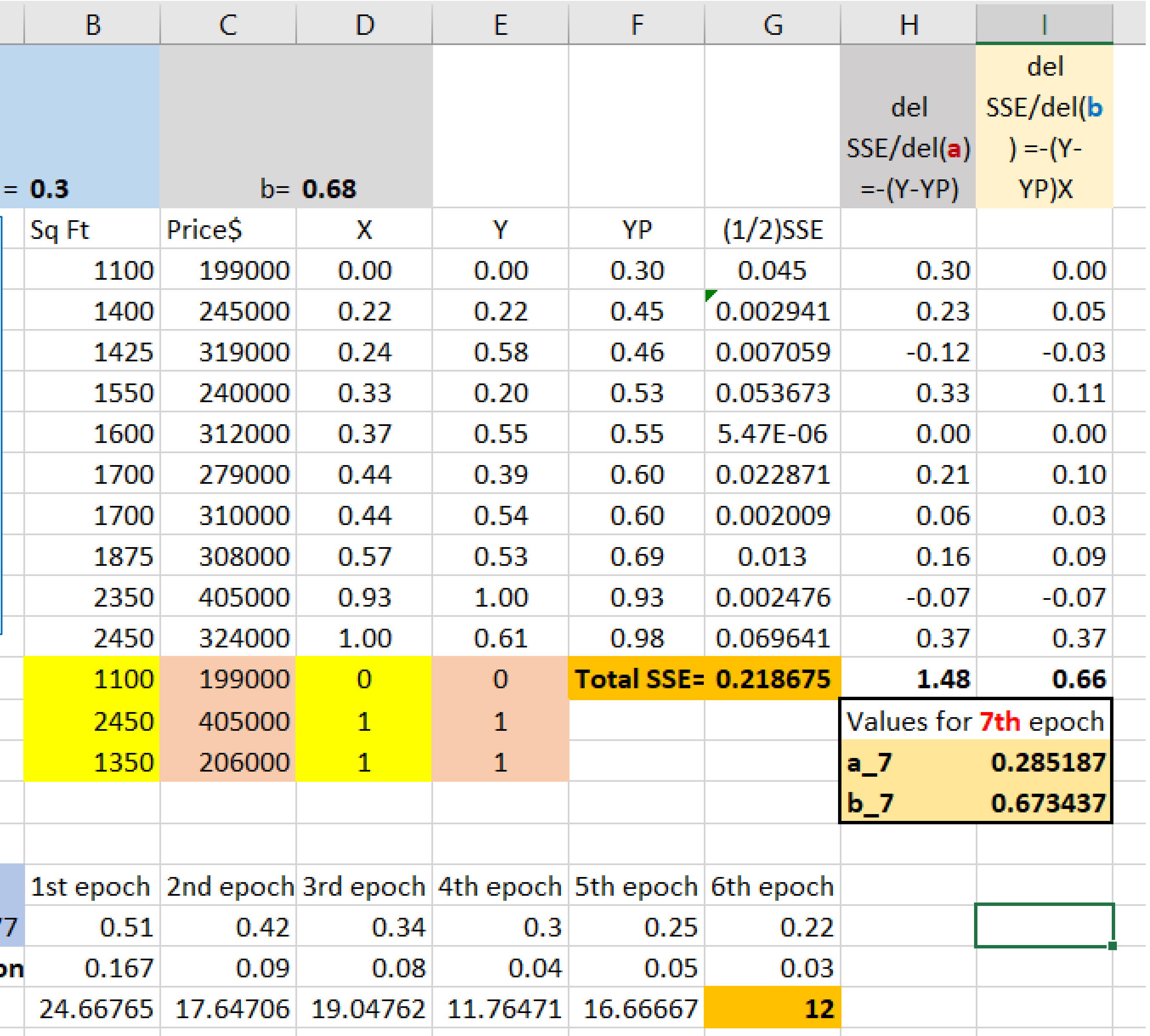
	A	B	C	D	E	F	G	H	I
1		a = **0.32**		b= **0.69**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.32	0.0512	0.32	0.00
		1400	245000	0.22	0.22	0.47	0.004675	0.25	0.06
		1425	319000	0.24	0.58	0.49	0.004648	-0.10	-0.02
		1550	240000	0.33	0.20	0.55	0.06159	0.35	0.12
		1600	312000	0.37	0.55	0.58	0.000365	0.03	0.01
		1700	279000	0.44	0.39	0.63	0.028398	0.24	0.11
		1700	310000	0.44	0.54	0.63	0.003857	0.09	0.04
		1875	308000	0.57	0.53	0.72	0.017482	0.19	0.11
		2350	405000	0.93	1.00	0.96	0.000845	-0.04	-0.04
		2450	324000	1.00	0.61	1.01	0.081287	0.40	0.40
13	MIN	1100	199000	0	0	**Total SSE=**	**0.254346**	**1.73**	**0.78**
14	MAX	2450	405000	1	1			Values for sixth epoch	
15	RANGE	1350	206000	1	1			**a_6**	**0.302732**
16								**b_6**	**0.682232**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch	5th epoch			
19	0.677	0.51	0.42	0.34	0.3	0.25			
20	**Reduction**	0.167	0.09	0.08	0.04	0.05			
21	**Redcn%**	24.66765	17.64706	19.04762	11.76471	**16.66667**			



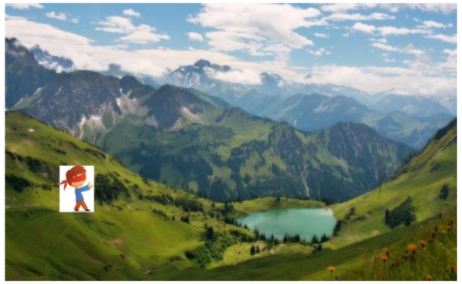
6^{th} epoch



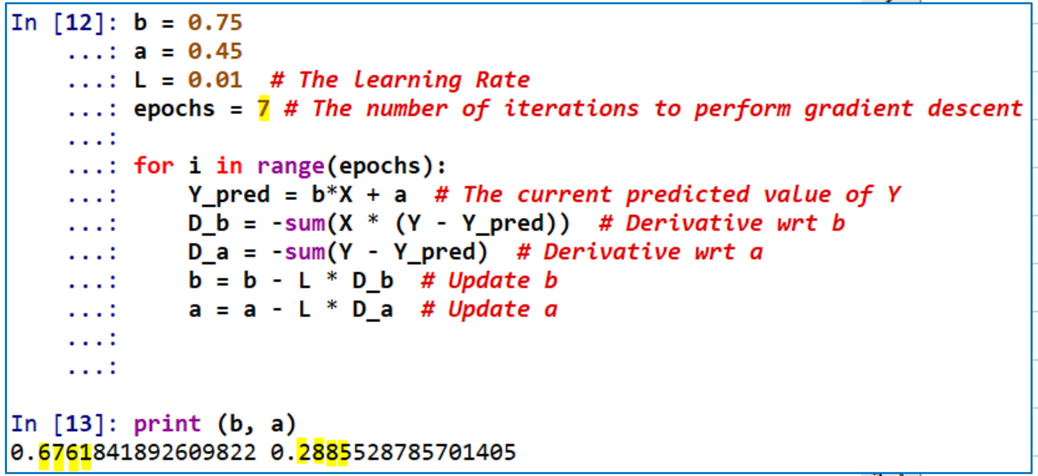
```
In [10]: b = 0.75
    ...: a = 0.45
    ...: L = 0.01  # The learning Rate
    ...: epochs = 6 # The number of iterations to perform gradient descent
    ...:
    ...: for i in range(epochs):
    ...:     Y_pred = b*X + a  # The current predicted value of Y
    ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
    ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
    ...:     b = b - L * D_b  # Update b
    ...:     a = a - L * D_a  # Update a
    ...:
    ...:

In [11]: print (b, a)
0.6828899663397007 0.30372898115773656
```



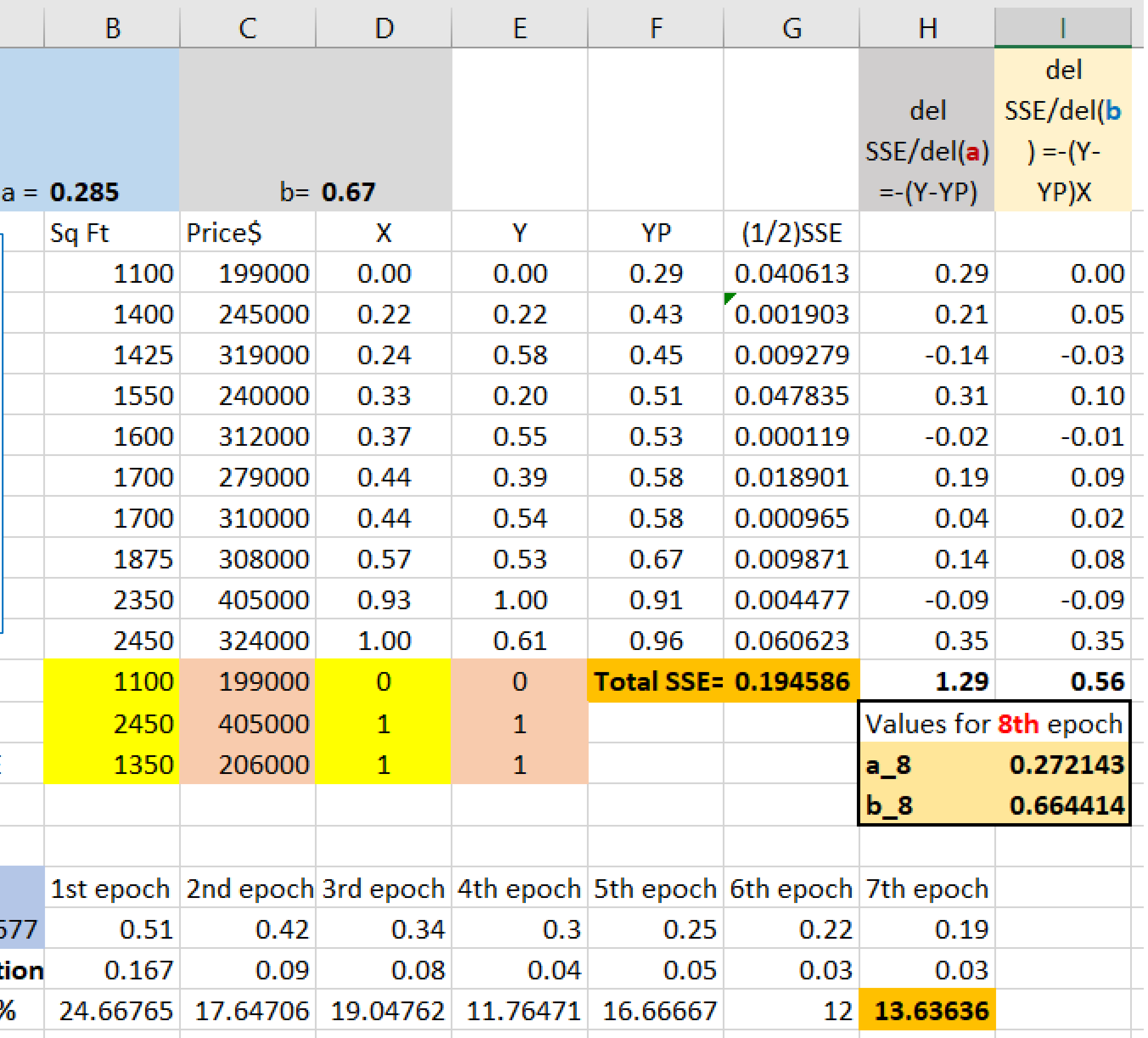
	A	B	C	D	E	F	G	H	I
1	a = **0.3**		b= **0.68**					del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.30	0.045	0.30	0.00
		1400	245000	0.22	0.22	0.45	0.002941	0.23	0.05
		1425	319000	0.24	0.58	0.46	0.007059	-0.12	-0.03
		1550	240000	0.33	0.20	0.53	0.053673	0.33	0.11
		1600	312000	0.37	0.55	0.55	5.47E-06	0.00	0.00
		1700	279000	0.44	0.39	0.60	0.022871	0.21	0.10
		1700	310000	0.44	0.54	0.60	0.002009	0.06	0.03
		1875	308000	0.57	0.53	0.69	0.013	0.16	0.09
		2350	405000	0.93	1.00	0.93	0.002476	-0.07	-0.07
12		2450	324000	1.00	0.61	0.98	0.069641	0.37	0.37
13	MIN	1100	199000	0	0	**Total SSE=**	**0.218675**	**1.48**	**0.66**
14	MAX	2450	405000	1	1			Values for 7th epoch	
15	RANGE	1350	206000	1	1			**a_7**	**0.285187**
16								**b_7**	**0.673437**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch	5th epoch	6th epoch		
19	0.677	0.51	0.42	0.34	0.3	0.25	0.22		
20	**Reduction**	0.167	0.09	0.08	0.04	0.05	0.03		
21	**Redcn%**	24.66765	17.64706	19.04762	11.76471	16.66667	**12**		



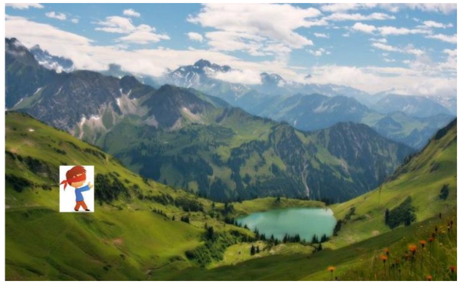
7th epoch



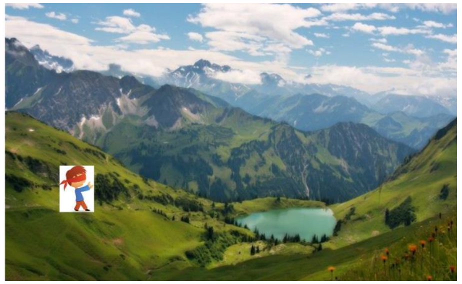
```
In [12]: b = 0.75
    ...: a = 0.45
    ...: L = 0.01  # The Learning Rate
    ...: epochs = 7 # The number of iterations to perform gradient descent
    ...:
    ...: for i in range(epochs):
    ...:     Y_pred = b*X + a  # The current predicted value of Y
    ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
    ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
    ...:     b = b - L * D_b  # Update b
    ...:     a = a - L * D_a  # Update a
    ...:
    ...:

In [13]: print (b, a)
0.6761841892609822 0.2885528785701405
```



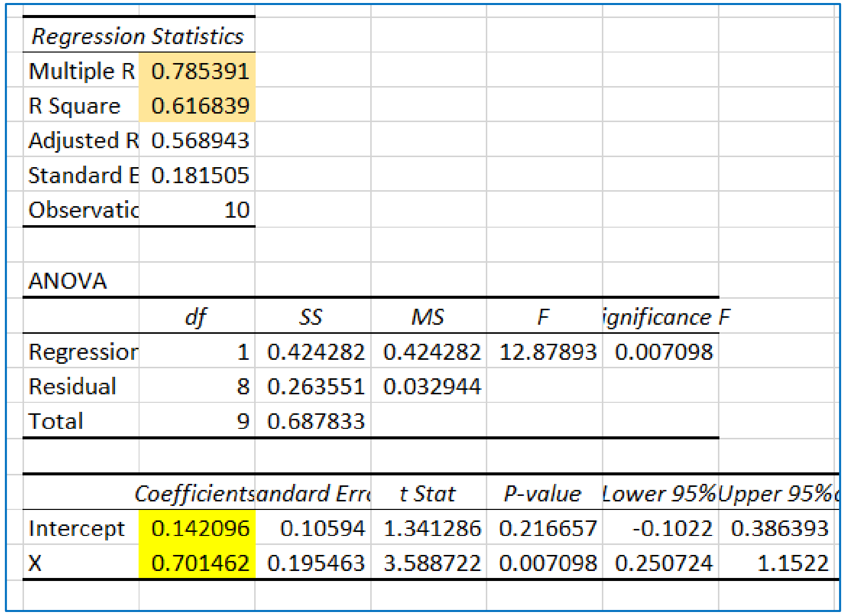
	A	B	C	D	E	F	G	H	I
1		a = **0.285**		b= **0.67**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
		1100	199000	0.00	0.00	0.29	0.040613	0.29	0.00
		1400	245000	0.22	0.22	0.43	0.001903	0.21	0.05
		1425	319000	0.24	0.58	0.45	0.009279	-0.14	-0.03
		1550	240000	0.33	0.20	0.51	0.047835	0.31	0.10
		1600	312000	0.37	0.55	0.53	0.000119	-0.02	-0.01
		1700	279000	0.44	0.39	0.58	0.018901	0.19	0.09
		1700	310000	0.44	0.54	0.58	0.000965	0.04	0.02
		1875	308000	0.57	0.53	0.67	0.009871	0.14	0.08
		2350	405000	0.93	1.00	0.91	0.004477	-0.09	-0.09
12		2450	324000	1.00	0.61	0.96	0.060623	0.35	0.35
13	MIN	1100	199000	0	0	**Total SSE=**	**0.194586**	**1.29**	**0.56**
14	MAX	2450	405000	1	1			Values for 8th epoch	
15	RANGE	1350	206000	1	1			**a_8**	**0.272143**
16								**b_8**	**0.664414**
17	SSE								
18	Org	1st epoch	2nd epoch	3rd epoch	4th epoch	5th epoch	6th epoch	7th epoch	
19	0.677	0.51	0.42	0.34	0.3	0.25	0.22	0.19	
20	**Reduction**	0.167	0.09	0.08	0.04	0.05	0.03	0.03	
21	**Redcn%**	24.66765	17.64706	19.04762	11.76471	16.66667	12	**13.63636**	



1000 epochs

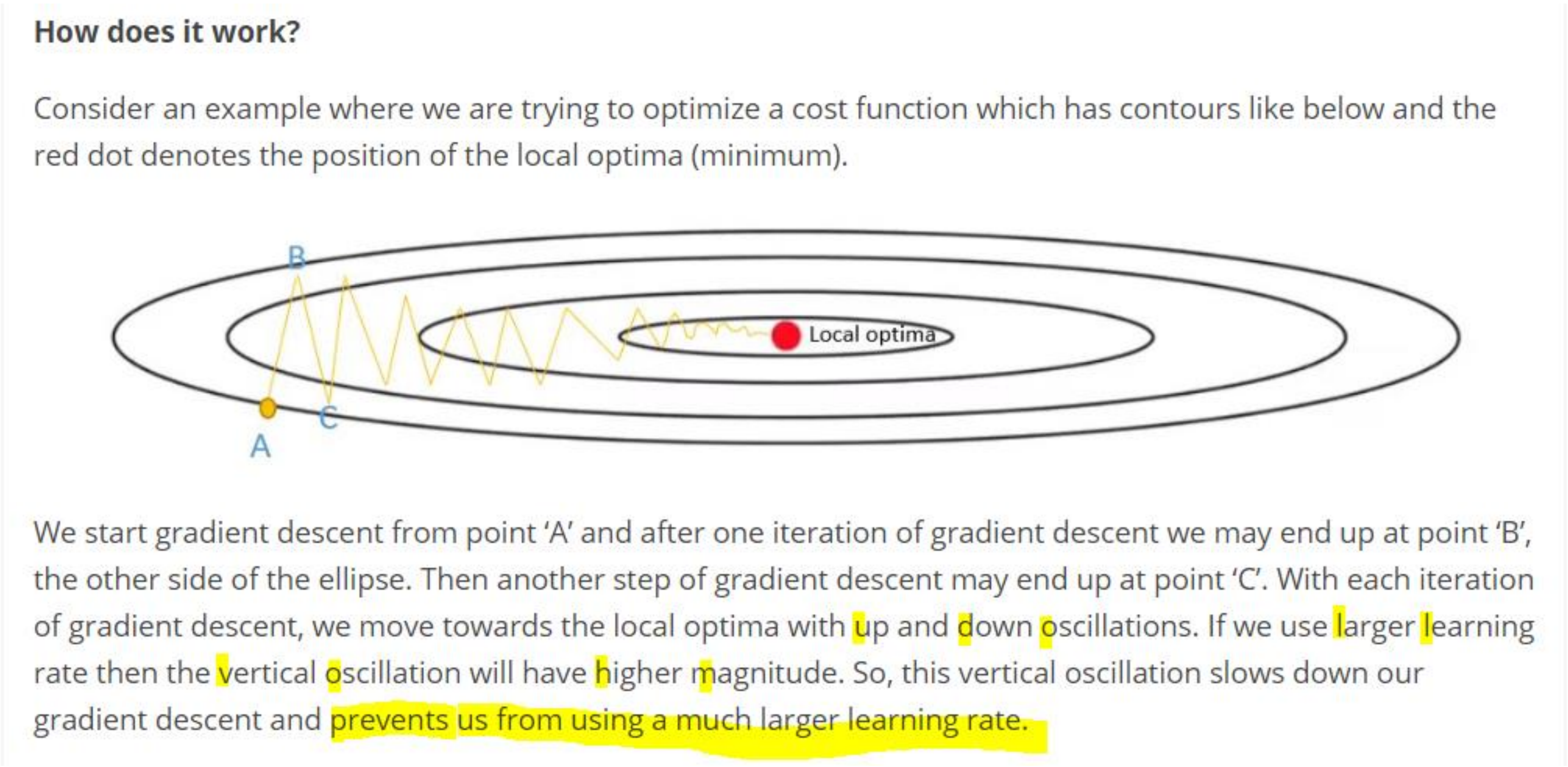


```
In [14]: b = 0.75
    ...: a = 0.45
    ...: L = 0.01  # The learning Rate
    ...: epochs = 1000 # The number of iterations to perform gradient descent
    ...:
    ...: for i in range(epochs):
    ...:     Y_pred = b*X + a  # The current predicted value of Y
    ...:     D_b = -sum(X * (Y - Y_pred))  # Derivative wrt b
    ...:     D_a = -sum(Y - Y_pred)  # Derivative wrt a
    ...:     b = b - L * D_b  # Update b
    ...:     a = a - L * D_a  # Update a
    ...:
    ...:

In [15]: print (b, a)
0.7040491329592683 0.14236404582620468
```



Regression Statistics						
Multiple R	0.785391					
R Square	0.616839					
Adjusted R	0.568943					
Standard E	0.181505					
Observatic	10					
ANOVA						
	df	*SS*	*MS*	*F*	*ignificance F*	
Regressior	1	0.424282	0.424282	12.87893	0.007098	
Residual	8	0.263551	0.032944			
Total	9	0.687833				
	Coefficients	*andard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*
Intercept	0.142096	0.10594	1.341286	0.216657	-0.1022	0.386393
X	0.701462	0.195463	3.588722	0.007098	0.250724	1.1522

https://engmrk.com/gradient-descent-with-momentum/

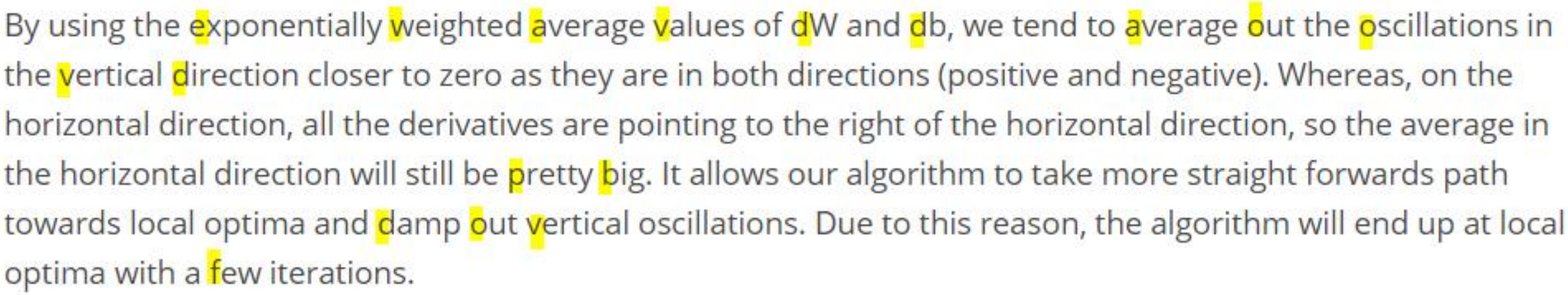


How does it work?

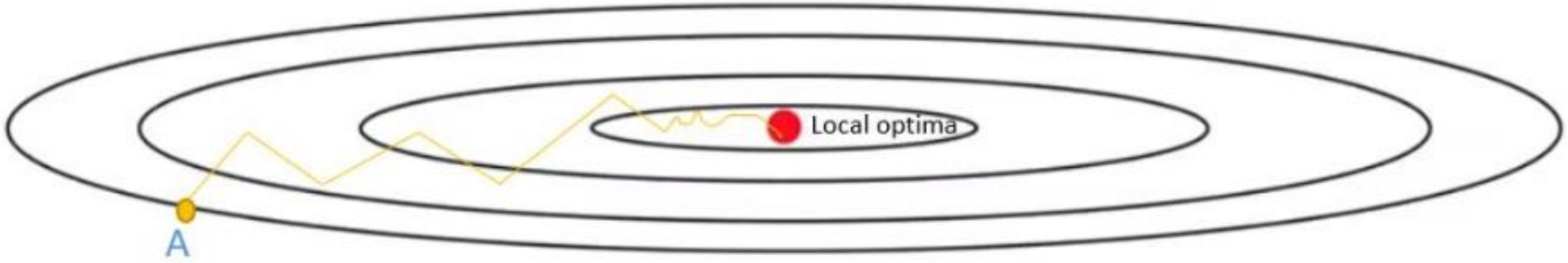
Consider an example where we are trying to optimize a cost function which has contours like below and the red dot denotes the position of the local optima (minimum).

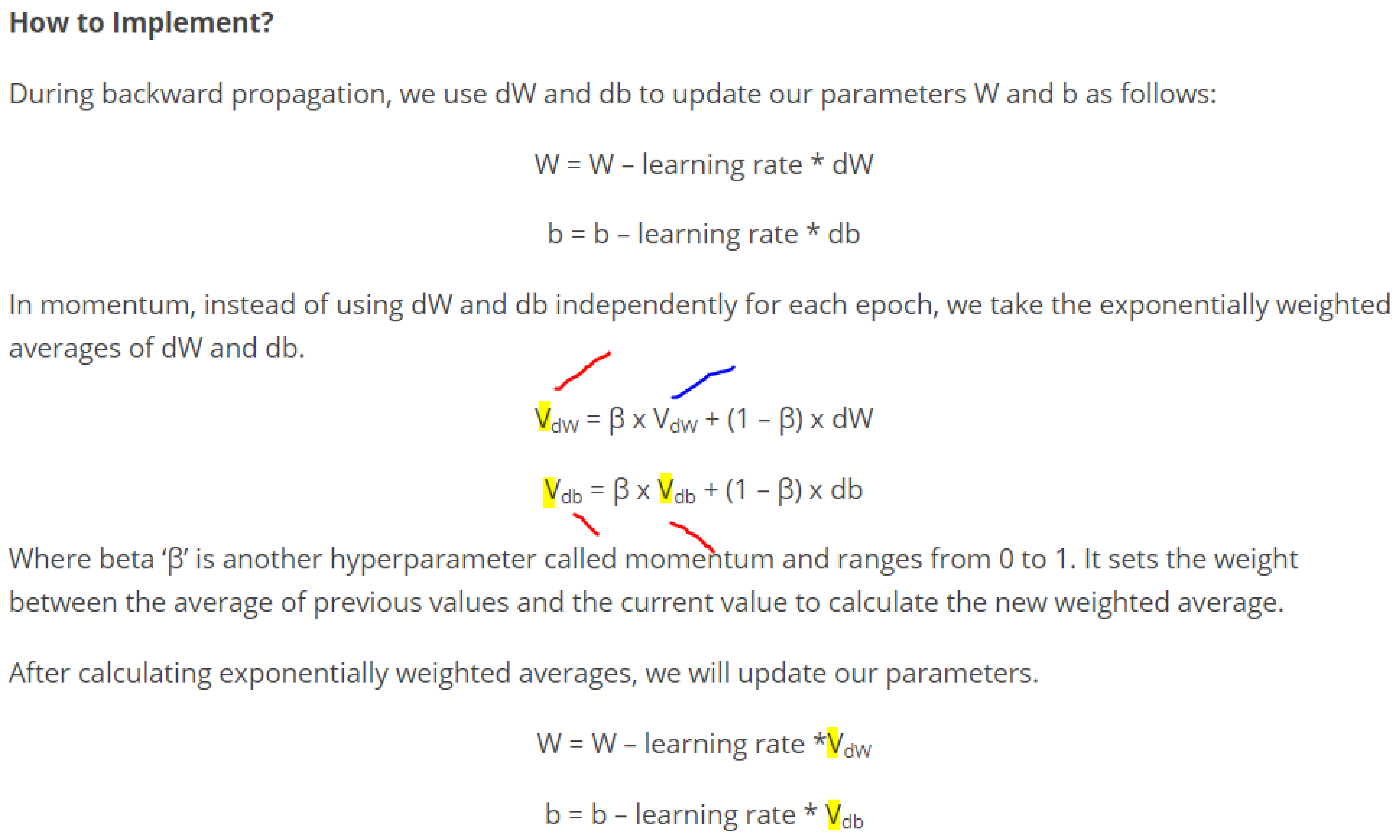
We start gradient descent from point 'A' and after one iteration of gradient descent we may end up at point 'B', the other side of the ellipse. Then another step of gradient descent may end up at point 'C'. With each iteration of gradient descent, we move towards the local optima with up and down oscillations. If we use larger learning rate then the vertical oscillation will have higher magnitude. So, this vertical oscillation slows down our gradient descent and prevents us from using a much larger learning rate.

Momentum



By using the exponentially weighted average values of dW and db, we tend to average out the oscillations in the vertical direction closer to zero as they are in both directions (positive and negative). Whereas, on the horizontal direction, all the derivatives are pointing to the right of the horizontal direction, so the average in the horizontal direction will still be pretty big. It allows our algorithm to take more straight forwards path towards local optima and damp out vertical oscillations. Due to this reason, the algorithm will end up at local optima with a few iterations.





How to Implement?

During backward propagation, we use dW and db to update our parameters W and b as follows:

$$W = W - \text{learning rate} * dW$$

$$b = b - \text{learning rate} * db$$

In momentum, instead of using dW and db independently for each epoch, we take the exponentially weighted averages of dW and db.

$$V_{dW} = \beta \times V_{dW} + (1 - \beta) \times dW$$

$$V_{db} = \beta \times V_{db} + (1 - \beta) \times db$$

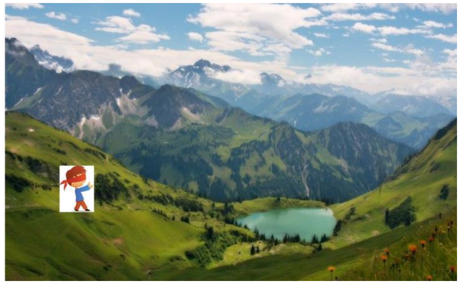
Where beta 'β' is another hyperparameter called momentum and ranges from 0 to 1. It sets the weight between the average of previous values and the current value to calculate the new weighted average.

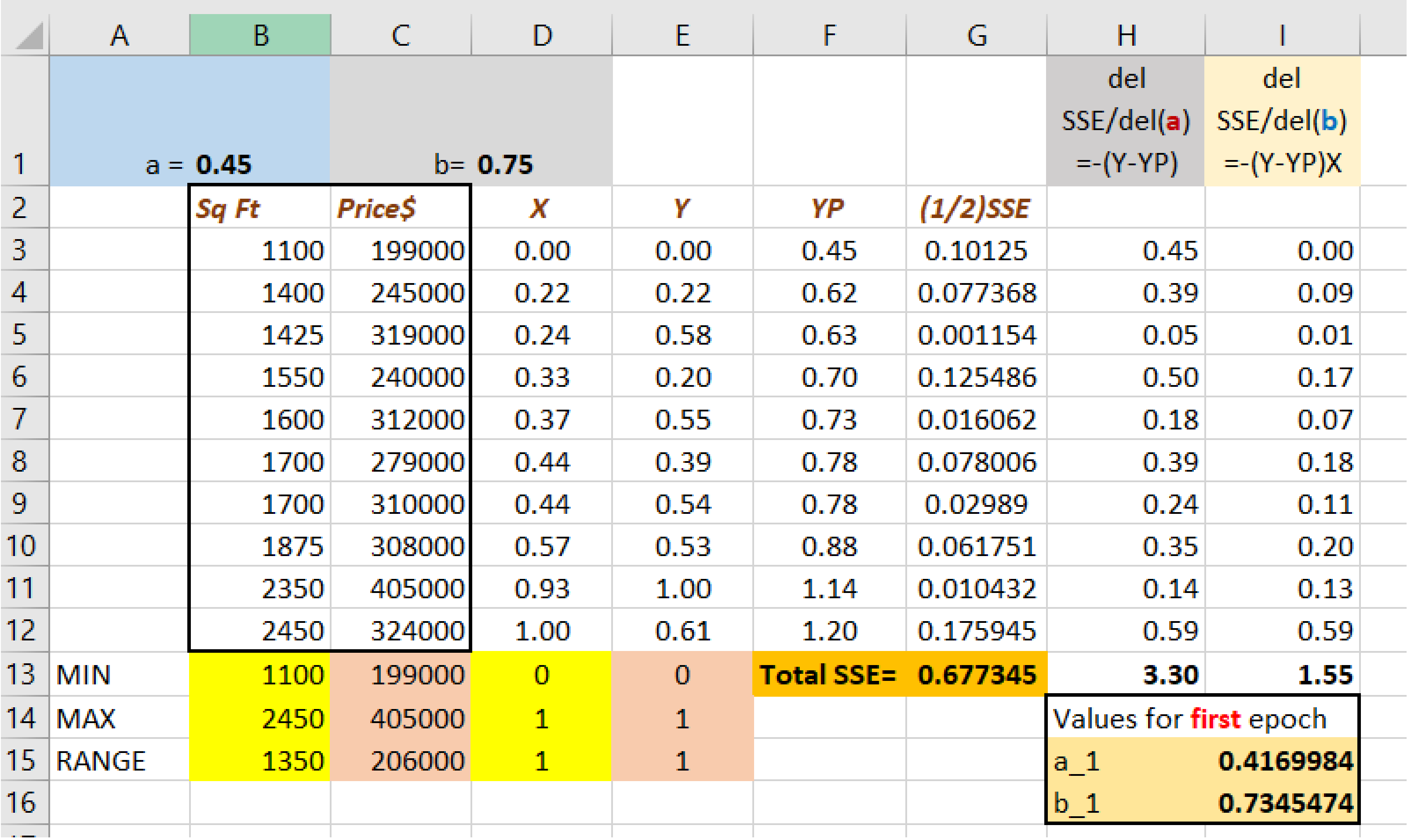
After calculating exponentially weighted averages, we will update our parameters.

$$W = W - \text{learning rate} * V_{dW}$$

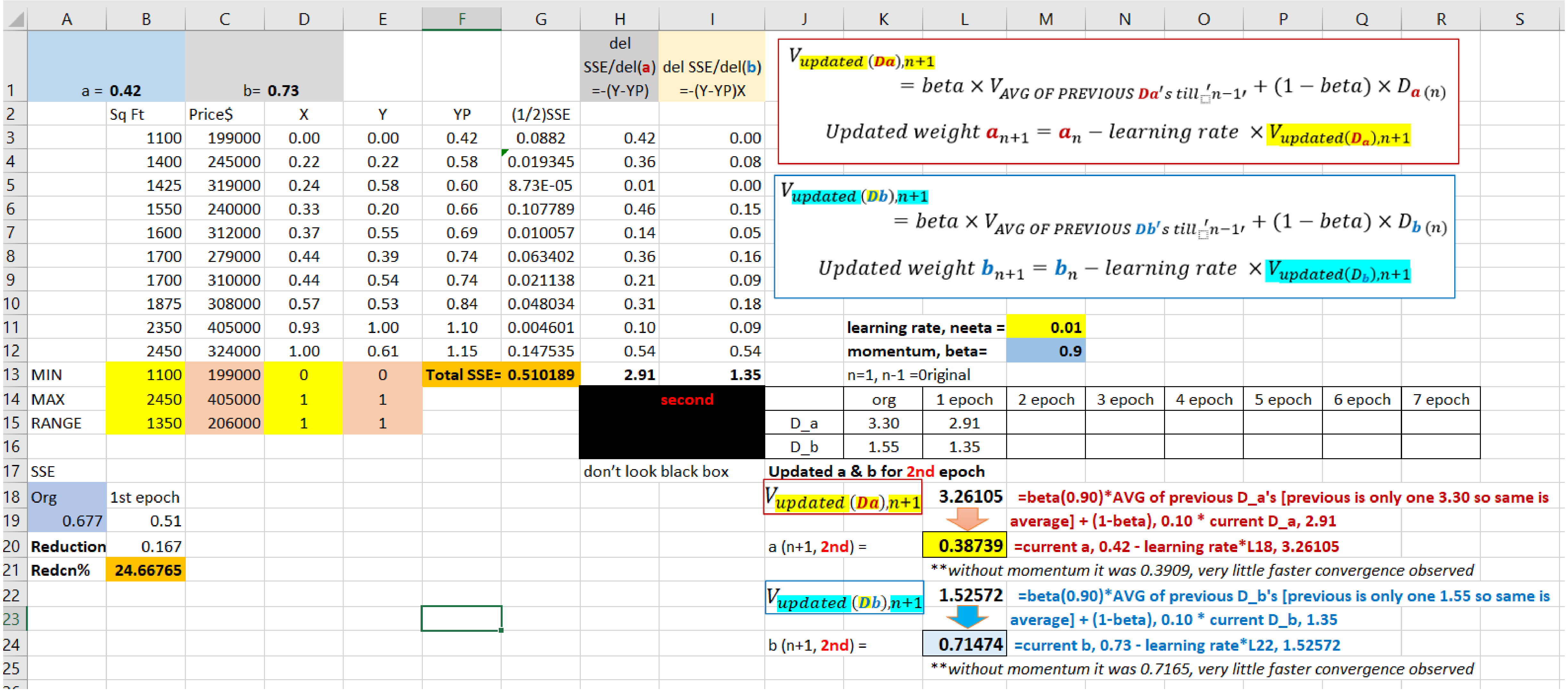
$$b = b - \text{learning rate} * V_{db}$$

Momentum





	A	B	C	D	E	F	G	H	I
1	a =	**0.45**	b=	**0.75**				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		***Sq Ft***	***Price$***	***X***	***Y***	***YP***	***(1/2)SSE***		
3		1100	199000	0.00	0.00	0.45	0.10125	0.45	0.00
4		1400	245000	0.22	0.22	0.62	0.077368	0.39	0.09
5		1425	319000	0.24	0.58	0.63	0.001154	0.05	0.01
6		1550	240000	0.33	0.20	0.70	0.125486	0.50	0.17
7		1600	312000	0.37	0.55	0.73	0.016062	0.18	0.07
8		1700	279000	0.44	0.39	0.78	0.078006	0.39	0.18
9		1700	310000	0.44	0.54	0.78	0.02989	0.24	0.11
10		1875	308000	0.57	0.53	0.88	0.061751	0.35	0.20
11		2350	405000	0.93	1.00	1.14	0.010432	0.14	0.13
12		2450	324000	1.00	0.61	1.20	0.175945	0.59	0.59
13	MIN	1100	199000	0	0	**Total SSE=**	**0.677345**	**3.30**	**1.55**
14	MAX	2450	405000	1	1			Values for **first** epoch	
15	RANGE	1350	206000	1	1			a_1	**0.4169984**
16								b_1	**0.7345474**



	A	B	C	D	E	F	G	H	I
1	a = **0.42**		b= **0.73**					del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.42	0.0882	0.42	0.00
4		1400	245000	0.22	0.22	0.58	0.019345	0.36	0.08
5		1425	319000	0.24	0.58	0.60	8.73E-05	0.01	0.00
6		1550	240000	0.33	0.20	0.66	0.107789	0.46	0.15
7		1600	312000	0.37	0.55	0.69	0.010057	0.14	0.05
8		1700	279000	0.44	0.39	0.74	0.063402	0.36	0.16
9		1700	310000	0.44	0.54	0.74	0.021138	0.21	0.09
10		1875	308000	0.57	0.53	0.84	0.048034	0.31	0.18
11		2350	405000	0.93	1.00	1.10	0.004601	0.10	0.09
12		2450	324000	1.00	0.61	1.15	0.147535	0.54	0.54
13	MIN	1100	199000	0	0	**Total SSE=**	**0.510189**	**2.91**	**1.35**
14	MAX	2450	405000	1	1			second	
15	RANGE	1350	206000	1	1				
16									
17	SSE							don't look black box	
18	Org	1st epoch							
19	0.677	0.51							
20	**Reduction**	0.167							
21	**Redcn%**	**24.66765**							

$$V_{updated\,(Da),n+1} = beta \times V_{AVG\ OF\ PREVIOUS\ Da's\ till\ 'n-1'} + (1 - beta) \times D_{a\,(n)}$$

$$Updated\ weight\ a_{n+1} = a_n - learning\ rate\ \times V_{updated(D_a),n+1}$$

$$V_{updated\,(Db),n+1} = beta \times V_{AVG\ OF\ PREVIOUS\ Db's\ till\ 'n-1'} + (1 - beta) \times D_{b\,(n)}$$

$$Updated\ weight\ b_{n+1} = b_n - learning\ rate\ \times V_{updated(D_b),n+1}$$

learning rate, neeta = **0.01**

momentum, beta= **0.9**

n=1, n-1 =0riginal

	org	1 epoch	2 epoch	3 epoch	4 epoch	5 epoch	6 epoch	7 epoch
D_a	3.30	2.91						
D_b	1.55	1.35						

Updated a & b for 2nd epoch

$V_{updated\,(Da),n+1}$ **3.26105** **=beta(0.90)*AVG of previous D_a's [previous is only one 3.30 so same is average] + (1-beta), 0.10 * current D_a, 2.91**

a (n+1, 2nd) = **0.38739** **=current a, 0.42 - learning rate*L18, 3.26105**

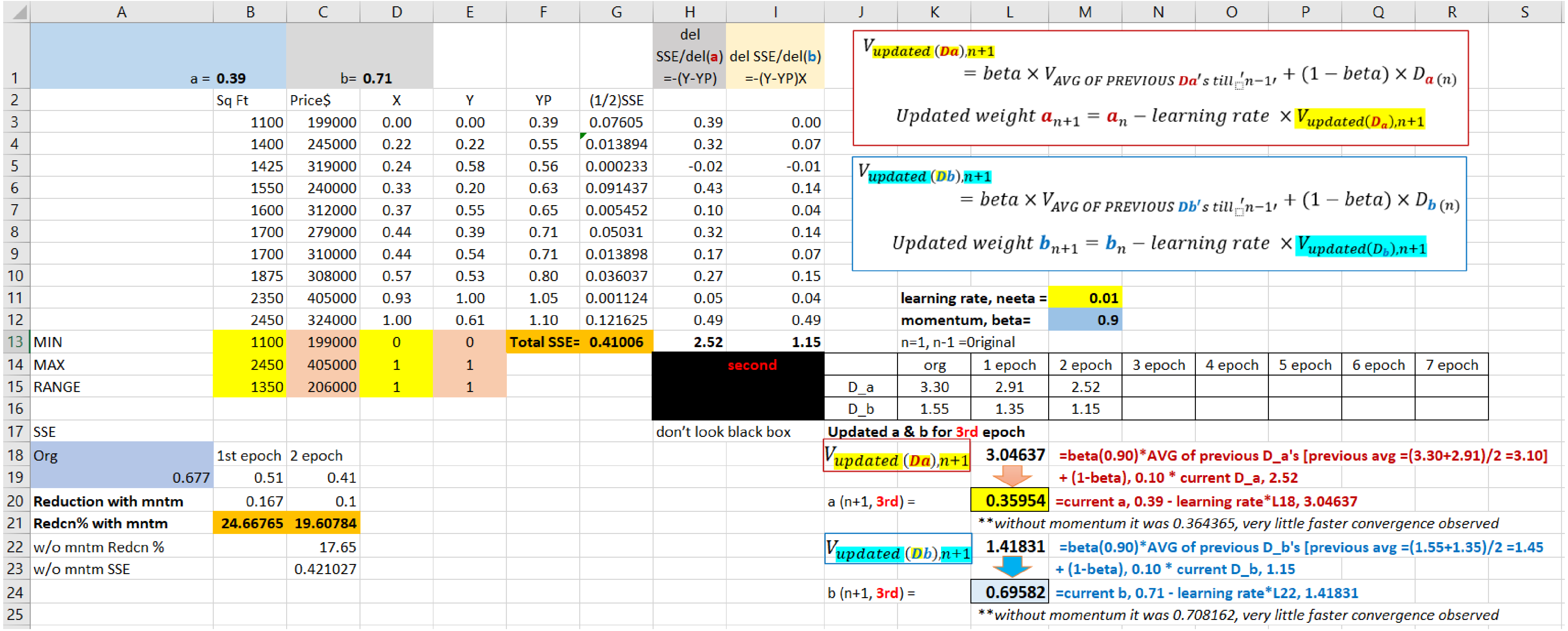
****without momentum it was 0.3909, very little faster convergence observed*

$V_{updated\,(Db),n+1}$ **1.52572** **=beta(0.90)*AVG of previous D_b's [previous is only one 1.55 so same is average] + (1-beta), 0.10 * current D_b, 1.35**

b (n+1, 2nd) = **0.71474** **=current b, 0.73 - learning rate*L22, 1.52572**

****without momentum it was 0.7165, very little faster convergence observed*

1st epoch with Momentum



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	a =	0.39	b=	0.71				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X										
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE												
3		1100	199000	0.00	0.00	0.39	0.07605	0.39	0.00										
4		1400	245000	0.22	0.22	0.55	0.013894	0.32	0.07										
5		1425	319000	0.24	0.58	0.56	0.000233	-0.02	-0.01										
6		1550	240000	0.33	0.20	0.63	0.091437	0.43	0.14										
7		1600	312000	0.37	0.55	0.65	0.005452	0.10	0.04										
8		1700	279000	0.44	0.39	0.71	0.05031	0.32	0.14										
9		1700	310000	0.44	0.54	0.71	0.013898	0.17	0.07										
10		1875	308000	0.57	0.53	0.80	0.036037	0.27	0.15										
11		2350	405000	0.93	1.00	1.05	0.001124	0.05	0.04		learning rate, neeta =		0.01						
12		2450	324000	1.00	0.61	1.10	0.121625	0.49	0.49		momentum, beta=		0.9						
13	MIN	1100	199000	0	0	Total SSE=	0.41006	2.52	1.15		n=1, n-1 =0riginal								
14	MAX	2450	405000	1	1			second			org	1 epoch	2 epoch	3 epoch	4 epoch	5 epoch	6 epoch	7 epoch	
15	RANGE	1350	206000	1	1			[redacted]	[redacted]	D_a	3.30	2.91	2.52						
16								[redacted]	[redacted]	D_b	1.55	1.35	1.15						
17	SSE							don't look black box		Updated a & b for 3rd epoch									
18	Org	1st epoch	2 epoch							$V_{updated\,(Da),n+1}$		3.04637	=beta(0.90)*AVG of previous D_a's [previous avg =(3.30+2.91)/2 =3.10]						
19	0.677	0.51	0.41										+ (1-beta), 0.10 * current D_a, 2.52						
20	Reduction with mntm	0.167	0.1							a (n+1, 3rd) =		0.35954	=current a, 0.39 - learning rate*L18, 3.04637						
21	Redcn% with mntm	24.66765	19.60784									**without momentum it was 0.364365, very little faster convergence observed							
22	w/o mntm Redcn %		17.65							$V_{updated\,(Db),n+1}$		1.41831	=beta(0.90)*AVG of previous D_b's [previous avg =(1.55+1.35)/2 =1.45						
23	w/o mntm SSE		0.421027										+ (1-beta), 0.10 * current D_b, 1.15						
24										b (n+1, 3rd) =		0.69582	=current b, 0.71 - learning rate*L22, 1.41831						
25												**without momentum it was 0.708162, very little faster convergence observed							

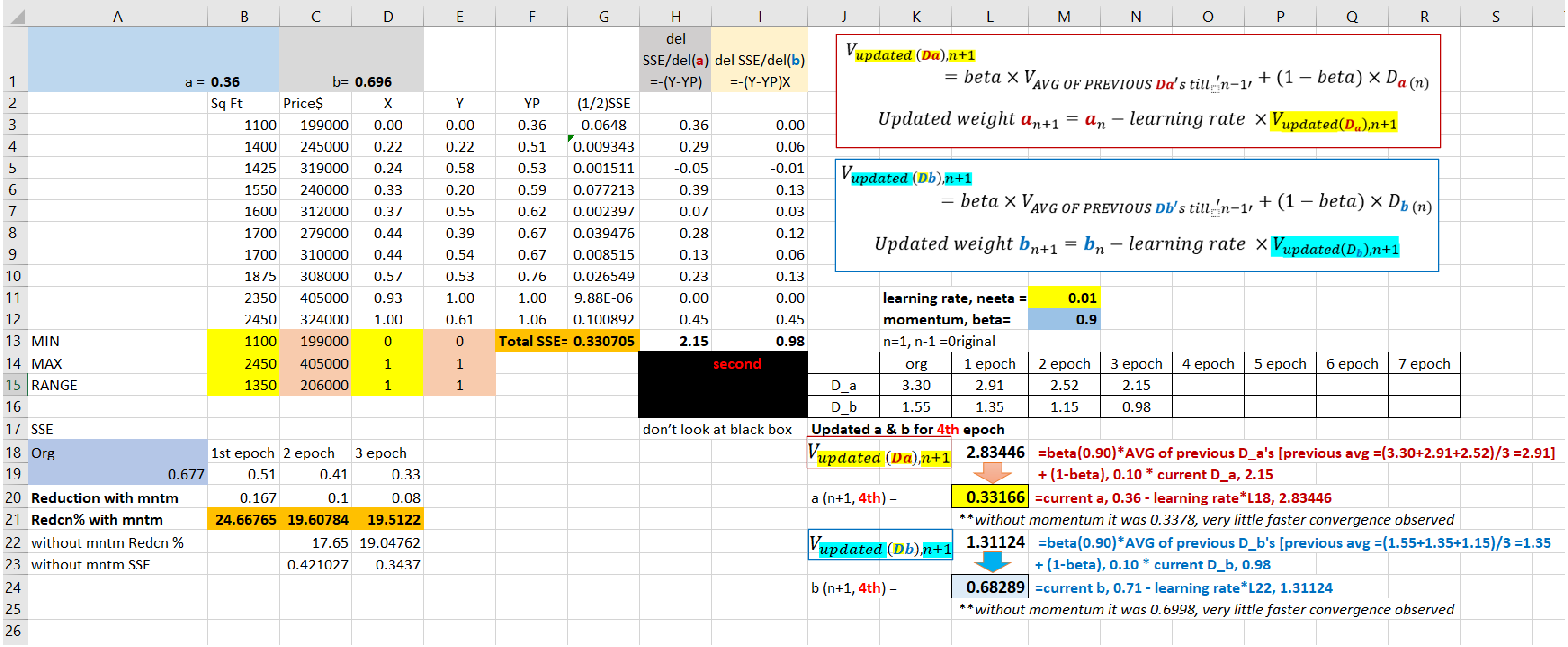
$$V_{updated\,(Da),n+1} = beta \times V_{AVG\ OF\ PREVIOUS\ Da's\ till\ 'n-1'} + (1 - beta) \times D_{a\,(n)}$$

$$Updated\ weight\ a_{n+1} = a_n - learning\ rate\ \times V_{updated(D_a),n+1}$$

$$V_{updated\,(Db),n+1} = beta \times V_{AVG\ OF\ PREVIOUS\ Db's\ till\ 'n-1'} + (1 - beta) \times D_{b\,(n)}$$

$$Updated\ weight\ b_{n+1} = b_n - learning\ rate\ \times V_{updated(D_b),n+1}$$

2nd epoch with Momentum



	A	B	C	D	E	F	G	H	I
1	a =	0.36	b=	0.696				del SSE/del(a) =-(Y-YP)	del SSE/del(b) =-(Y-YP)X
2		Sq Ft	Price$	X	Y	YP	(1/2)SSE		
3		1100	199000	0.00	0.00	0.36	0.0648	0.36	0.00
4		1400	245000	0.22	0.22	0.51	0.009343	0.29	0.06
5		1425	319000	0.24	0.58	0.53	0.001511	-0.05	-0.01
6		1550	240000	0.33	0.20	0.59	0.077213	0.39	0.13
7		1600	312000	0.37	0.55	0.62	0.002397	0.07	0.03
8		1700	279000	0.44	0.39	0.67	0.039476	0.28	0.12
9		1700	310000	0.44	0.54	0.67	0.008515	0.13	0.06
10		1875	308000	0.57	0.53	0.76	0.026549	0.23	0.13
11		2350	405000	0.93	1.00	1.00	9.88E-06	0.00	0.00
12		2450	324000	1.00	0.61	1.06	0.100892	0.45	0.45
13	MIN	1100	199000	0	0	Total SSE=	0.330705	2.15	0.98
14	MAX	2450	405000	1	1			second	
15	RANGE	1350	206000	1	1				
16									
17	SSE							don't look at black box	
18	Org	1st epoch	2 epoch	3 epoch					
19	0.677	0.51	0.41	0.33					
20	Reduction with mntm	0.167	0.1	0.08					
21	Redcn% with mntm	24.66765	19.60784	19.5122					
22	without mntm Redcn %		17.65	19.04762					
23	without mntm SSE		0.421027	0.3437					
24									
25									
26									

$$V_{updated\,(Da),n+1} = beta \times V_{AVG\ OF\ PREVIOUS\ Da's\ till\ 'n-1'} + (1 - beta) \times D_{a\,(n)}$$

$$Updated\ weight\ a_{n+1} = a_n - learning\ rate\ \times V_{updated(D_a),n+1}$$

$$V_{updated\,(Db),n+1} = beta \times V_{AVG\ OF\ PREVIOUS\ Db's\ till\ 'n-1'} + (1 - beta) \times D_{b\,(n)}$$

$$Updated\ weight\ b_{n+1} = b_n - learning\ rate\ \times V_{updated(D_b),n+1}$$

learning rate, neeta = **0.01**

momentum, beta= **0.9**

n=1, n-1 =0riginal

	org	1 epoch	2 epoch	3 epoch	4 epoch	5 epoch	6 epoch	7 epoch
D_a	3.30	2.91	2.52	2.15				
D_b	1.55	1.35	1.15	0.98				

Updated a & b for 4th epoch

$V_{updated\,(Da),n+1}$ **2.83446** **=beta(0.90)*AVG of previous D_a's [previous avg =(3.30+2.91+2.52)/3 =2.91]**
+ (1-beta), 0.10 * current D_a, 2.15

a (n+1, 4th) = **0.33166** **=current a, 0.36 - learning rate*L18, 2.83446**

****without momentum it was 0.3378, very little faster convergence observed*

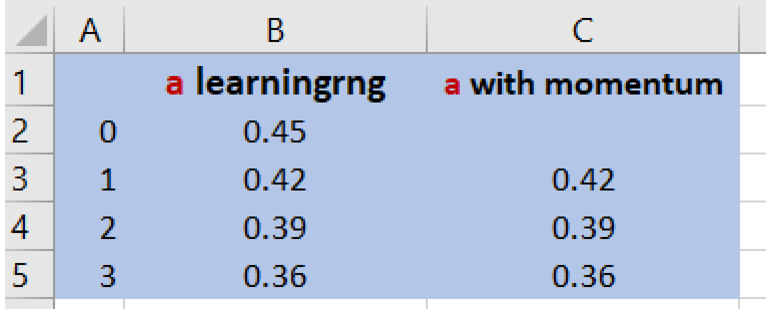
$V_{updated\,(Db),n+1}$ **1.31124** **=beta(0.90)*AVG of previous D_b's [previous avg =(1.55+1.35+1.15)/3 =1.35**
+ (1-beta), 0.10 * current D_b, 0.98

b (n+1, 4th) = **0.68289** **=current b, 0.71 - learning rate*L22, 1.31124**

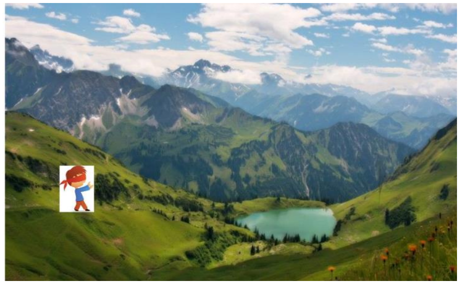
****without momentum it was 0.6998, very little faster convergence observed*

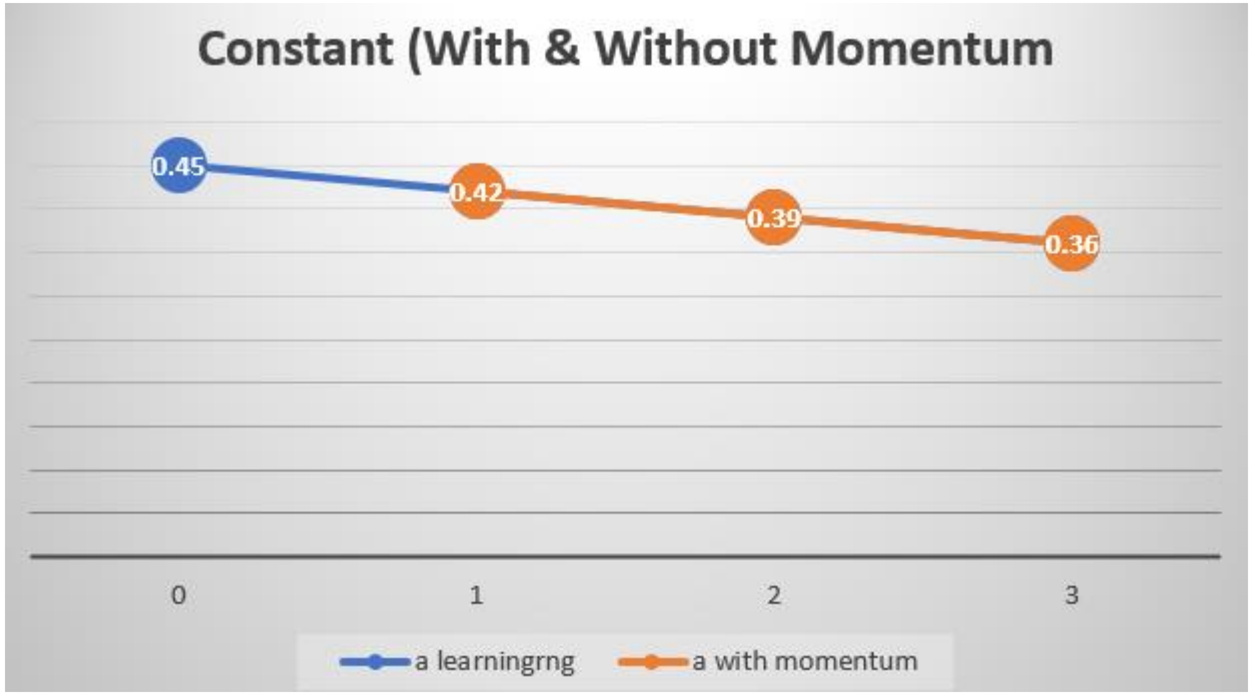
3rd epoch with Momentum

Comparison in ‘a’ {with & without Momentum}

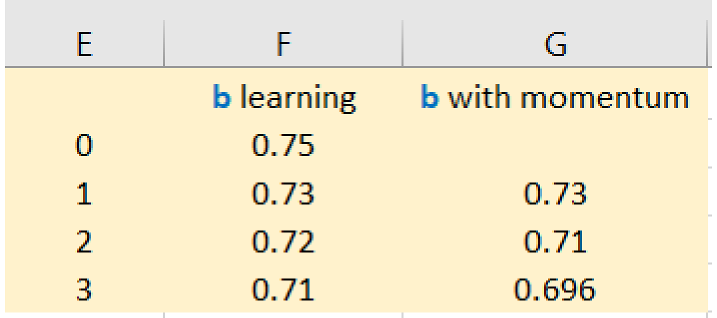


	A	B	C
1		a learningrng	a with momentum
2	0	0.45	
3	1	0.42	0.42
4	2	0.39	0.39
5	3	0.36	0.36

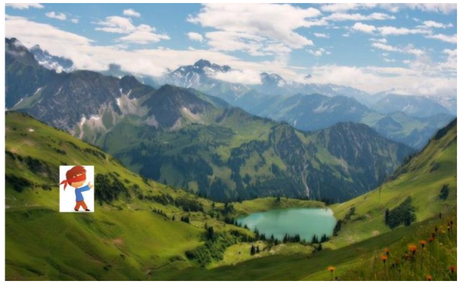


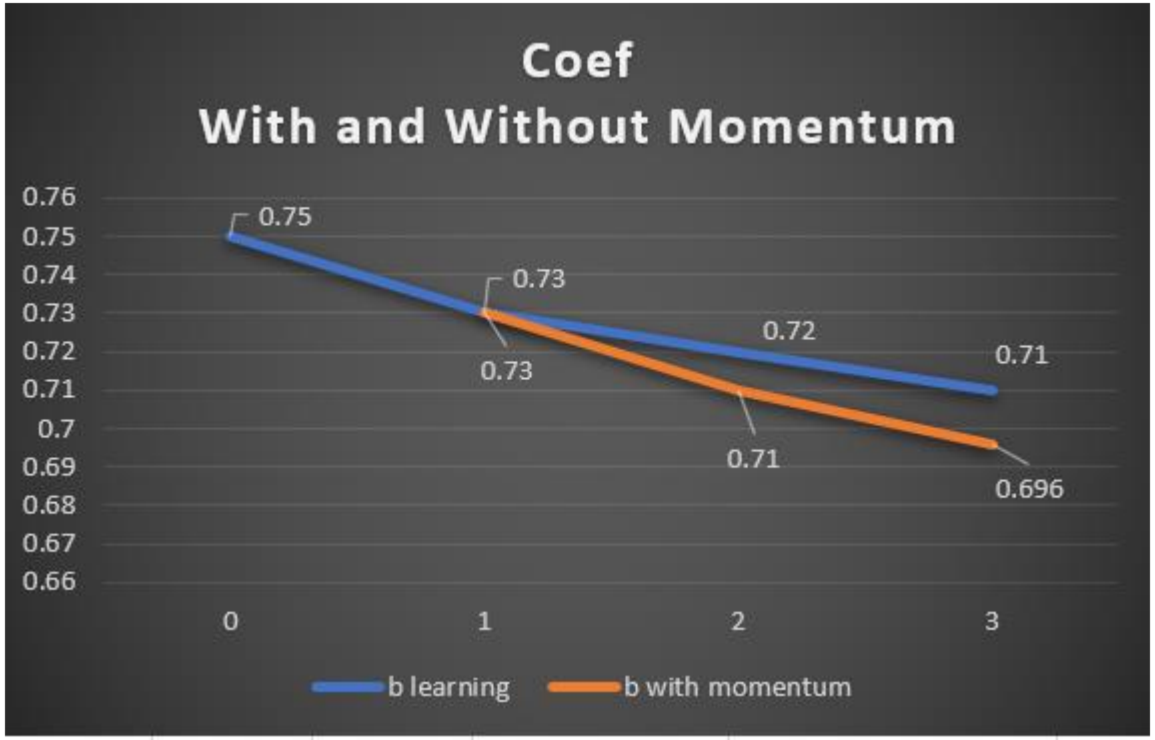


Comparison in ‘b’ {with & without Momentum}

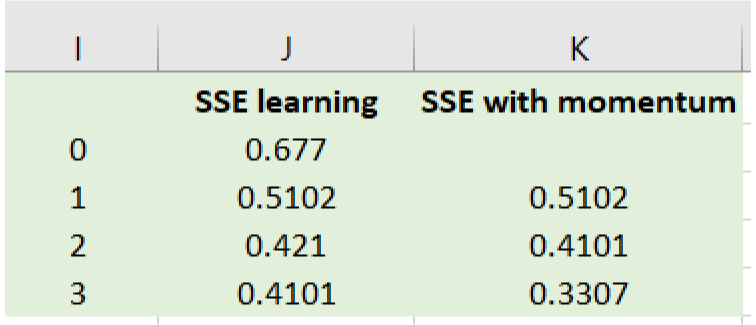


E	F	G
	b learning	b with momentum
0	0.75	
1	0.73	0.73
2	0.72	0.71
3	0.71	0.696

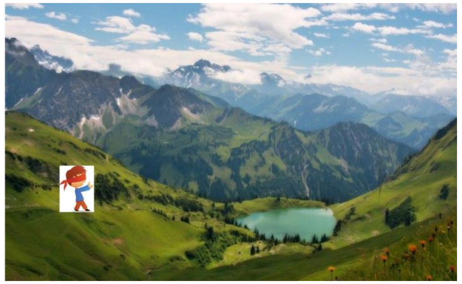


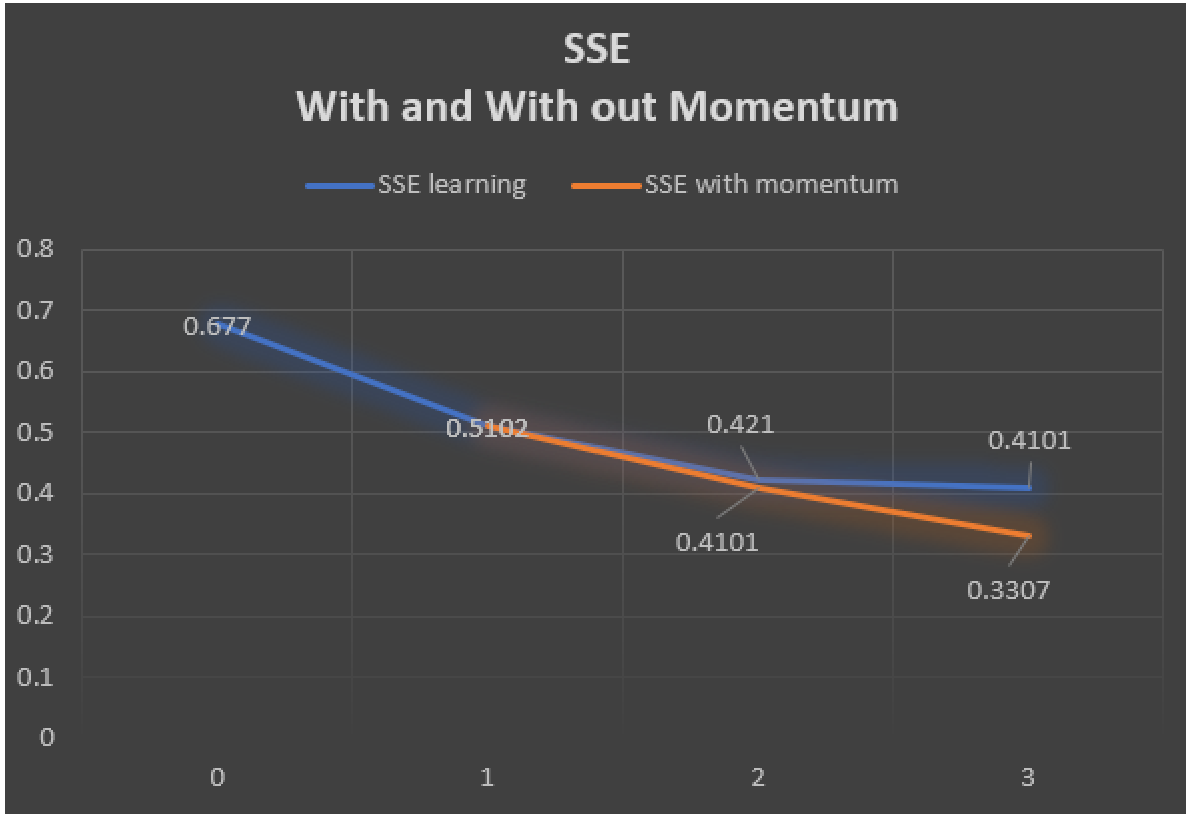


Comparison in 'SSE' {with & without Momentum}

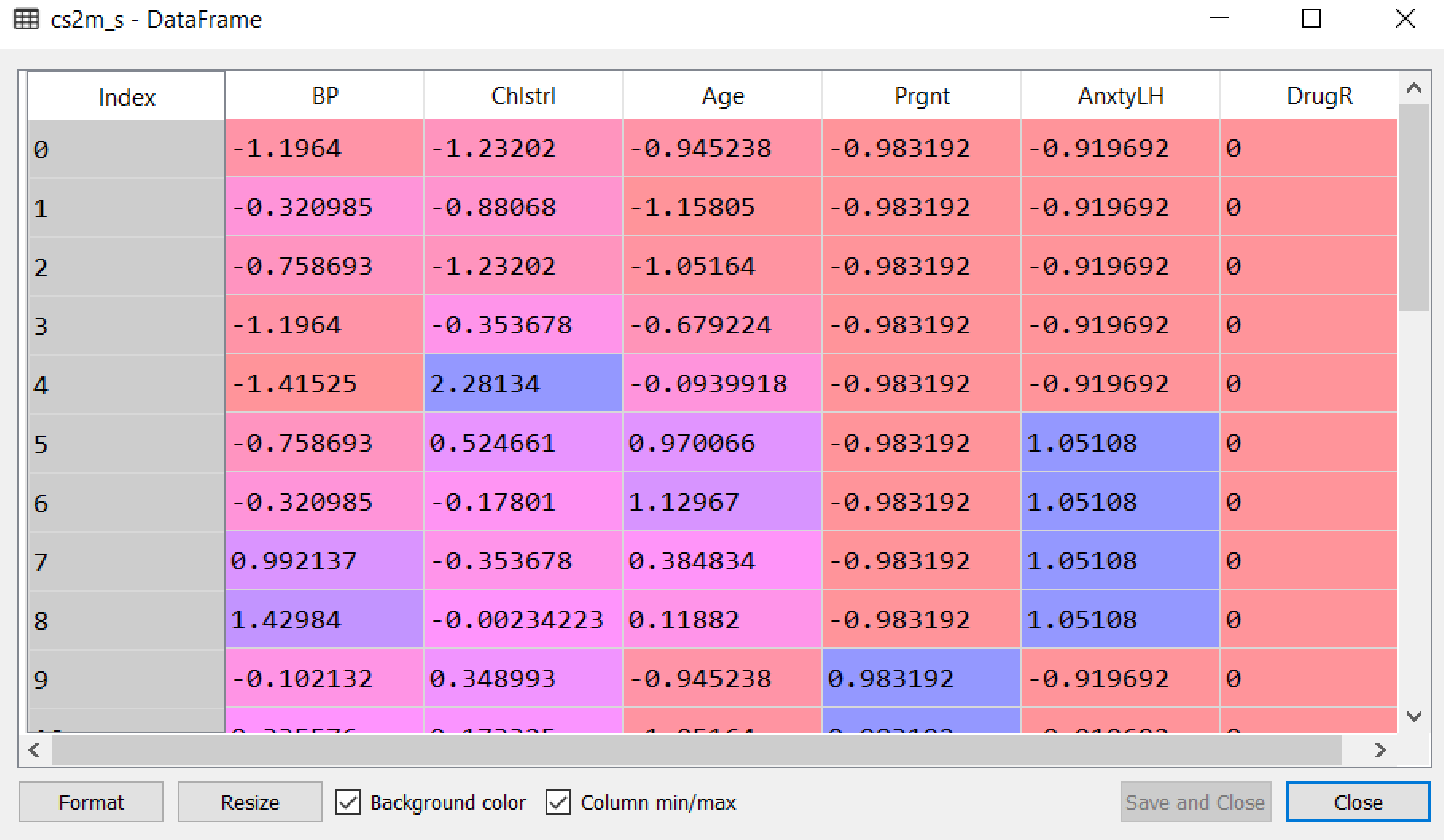


I	J	K
	SSE learning	SSE with momentum
0	0.677	
1	0.5102	0.5102
2	0.421	0.4101
3	0.4101	0.3307





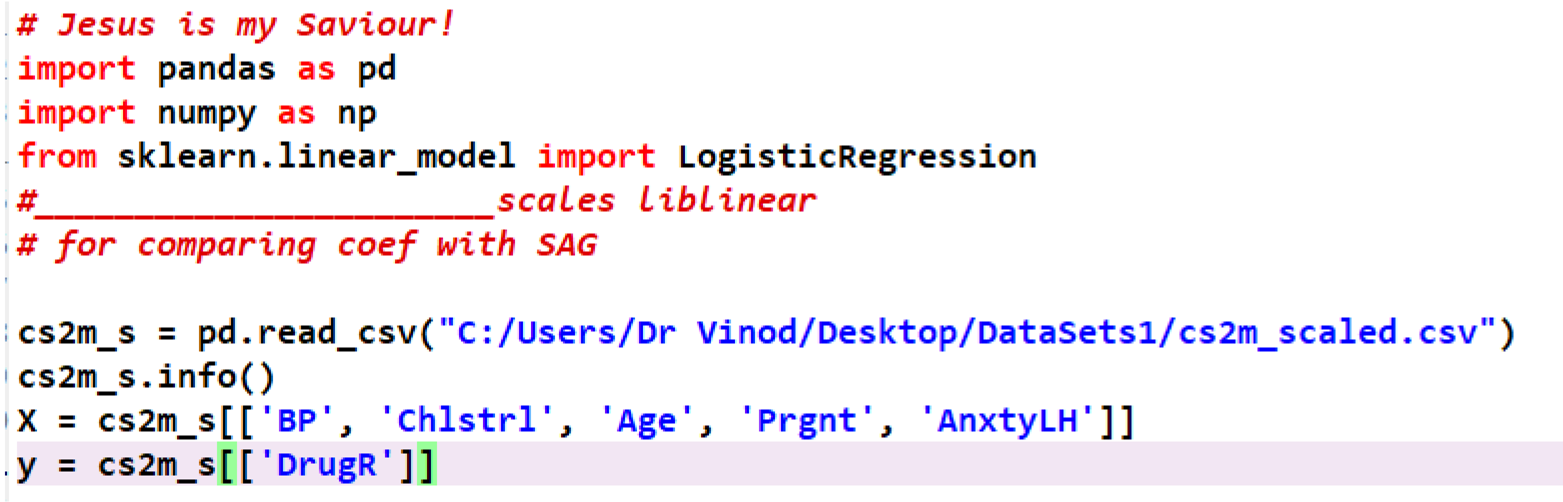
Logistic Regression



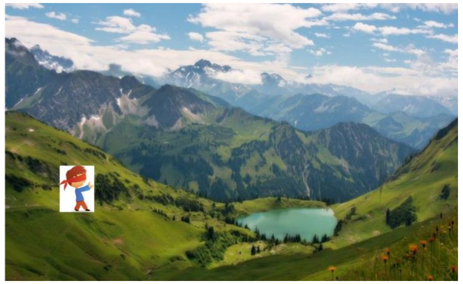
cs2m_s - DataFrame

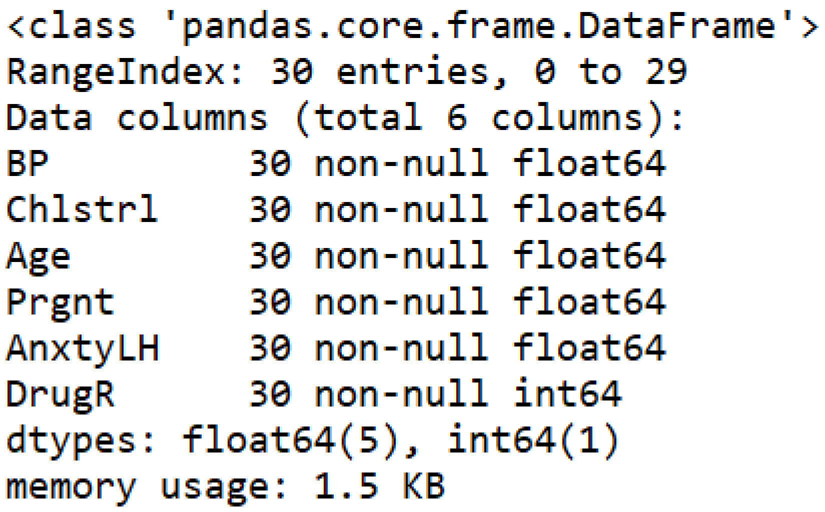
Index	BP	Chlstrl	Age	Prgnt	AnxtyLH	DrugR
0	-1.1964	-1.23202	-0.945238	-0.983192	-0.919692	0
1	-0.320985	-0.88068	-1.15805	-0.983192	-0.919692	0
2	-0.758693	-1.23202	-1.05164	-0.983192	-0.919692	0
3	-1.1964	-0.353678	-0.679224	-0.983192	-0.919692	0
4	-1.41525	2.28134	-0.0939918	-0.983192	-0.919692	0
5	-0.758693	0.524661	0.970066	-0.983192	1.05108	0
6	-0.320985	-0.17801	1.12967	-0.983192	1.05108	0
7	0.992137	-0.353678	0.384834	-0.983192	1.05108	0
8	1.42984	-0.00234223	0.11882	-0.983192	1.05108	0
9	-0.102132	0.348993	-0.945238	0.983192	-0.919692	0

Scaled Data



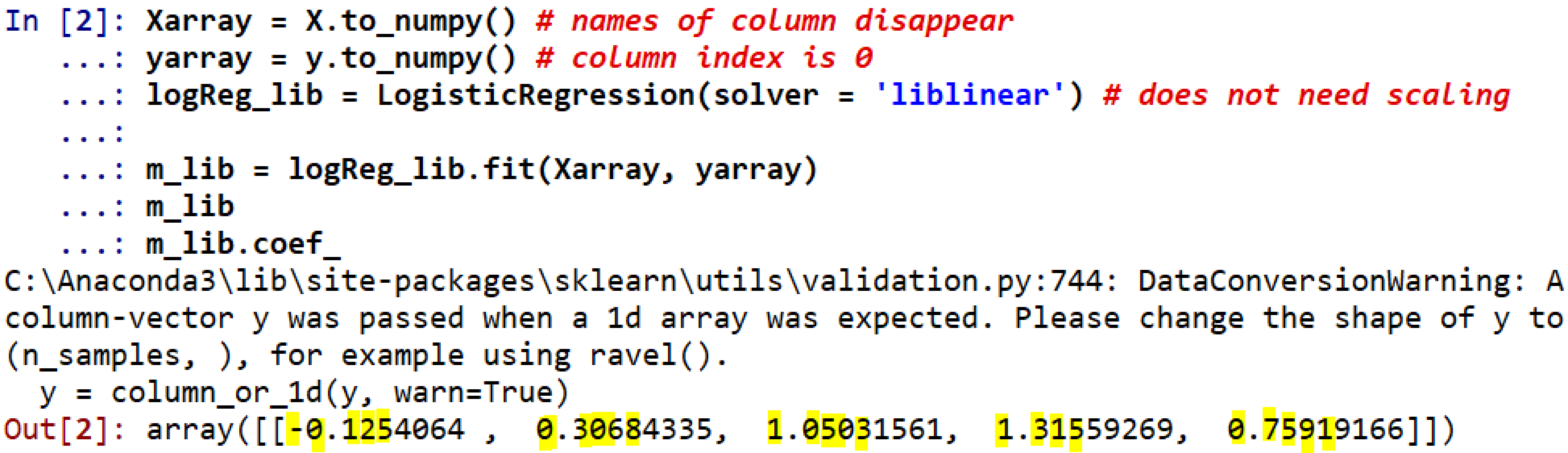
```
# Jesus is my Saviour!
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
#_______________________scales liblinear
# for comparing coef with SAG

cs2m_s = pd.read_csv("C:/Users/Dr Vinod/Desktop/DataSets1/cs2m_scaled.csv")
cs2m_s.info()
X = cs2m_s[['BP', 'Chlstrl', 'Age', 'Prgnt', 'AnxtyLH']]
y = cs2m_s[['DrugR']]
```



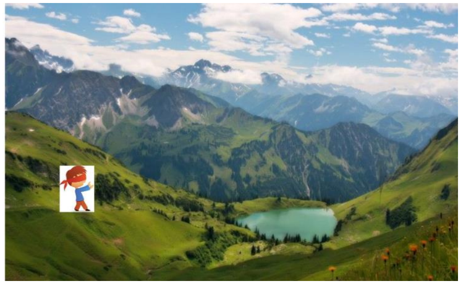


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
BP         30 non-null float64
Chlstrl    30 non-null float64
Age        30 non-null float64
Prgnt      30 non-null float64
AnxtyLH    30 non-null float64
DrugR      30 non-null int64
dtypes: float64(5), int64(1)
memory usage: 1.5 KB
```

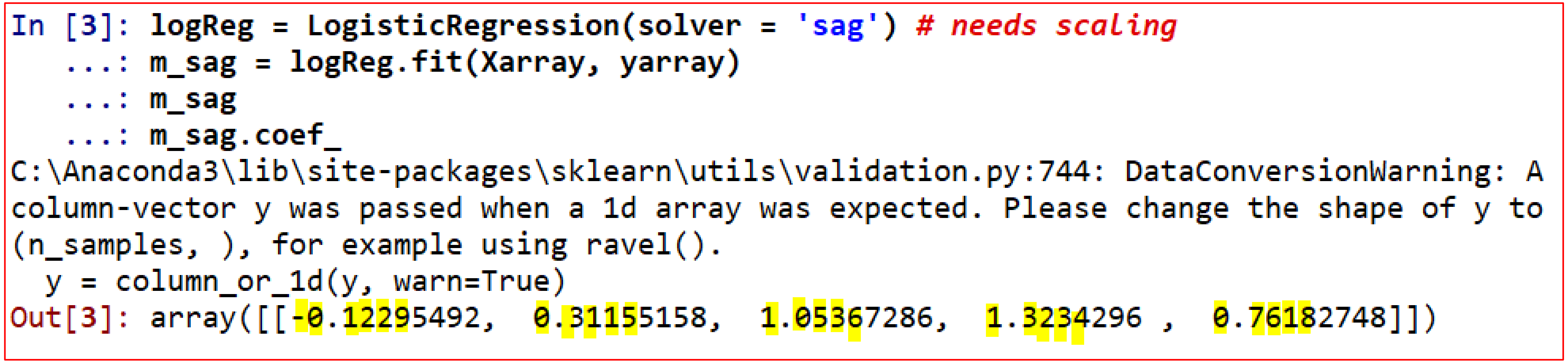
Liblinear solver



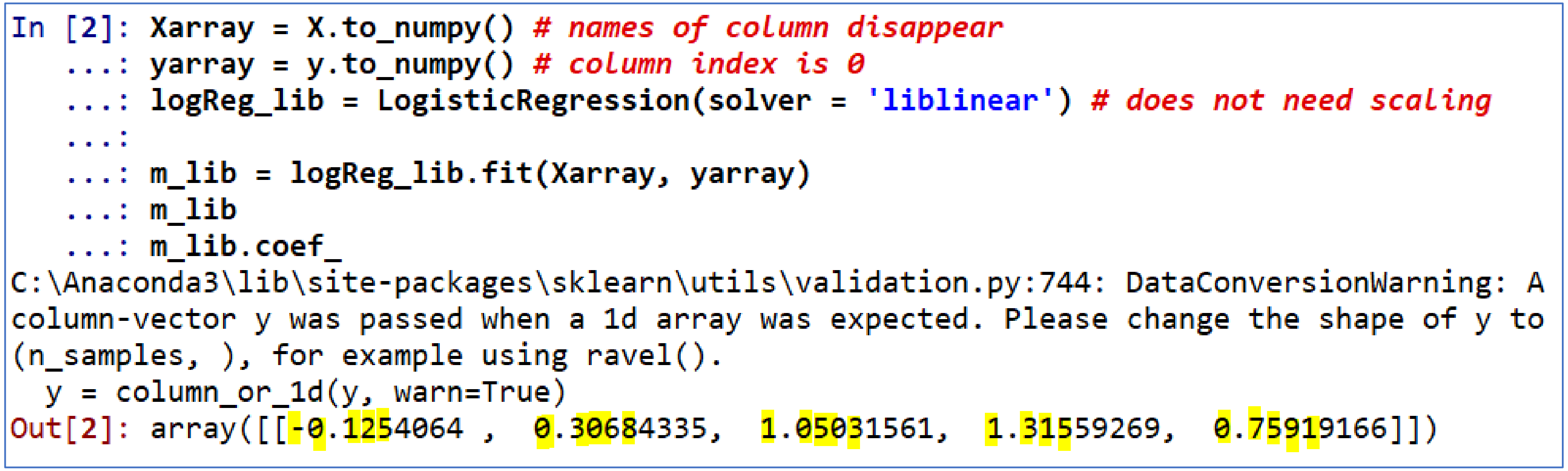
```
In [2]: Xarray = X.to_numpy() # names of column disappear
   ...: yarray = y.to_numpy() # column index is 0
   ...: logReg_lib = LogisticRegression(solver = 'liblinear') # does not need scaling
   ...:
   ...: m_lib = logReg_lib.fit(Xarray, yarray)
   ...: m_lib
   ...: m_lib.coef_
C:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:744: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Out[2]: array([[-0.1254064 ,  0.30684335,  1.05031561,  1.31559269,  0.75919166]])
```



Stochastic Gradient Descent



```
In [3]: logReg = LogisticRegression(solver = 'sag') # needs scaling
   ...: m_sag = logReg.fit(Xarray, yarray)
   ...: m_sag
   ...: m_sag.coef_
C:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:744: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Out[3]: array([[-0.12295492,  0.31155158,  1.05367286,  1.3234296 ,  0.76182748]])
```



```
In [2]: Xarray = X.to_numpy() # names of column disappear
   ...: yarray = y.to_numpy() # column index is 0
   ...: logReg_lib = LogisticRegression(solver = 'liblinear') # does not need scaling
   ...:
   ...: m_lib = logReg_lib.fit(Xarray, yarray)
   ...: m_lib
   ...: m_lib.coef_
C:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:744: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Out[2]: array([[-0.1254064 ,  0.30684335,  1.05031561,  1.31559269,  0.75919166]])
```

2 teach is
+
2 touch lives

4 ever